UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Operational Intelligence With GDPR

André Sequeira Borges Lourenço

Mestrado em Segurança Informática

Trabalho de Projeto orientado por:
Prof. Maria Dulce Pedroso Domingos
Nuno Alexandre Matias Gomes Dias

2018

Abstract:

More and more we live in a world that's powered by data. All kinds of applications, from social networks to business products, produce data just by existing and performing the actions that they're intended to. Take for example a telco company where thousands of both residential and business customer orders arrive each day, it is expected that at the very moment we buy a product we're giving away info about what we're buying and, more important, about who we are. It is of the company's interest that all data is kept in an interpretable format in order to infer what products interest their customers and analyse the company's operations to maintain their service as available and efficient as possible. These systems are normally decentralized by nature, meaning that it is not uncommon to have separate sub-systems with well-defined purposes, like Customer Relationship Management (CRM) systems that handle customer orders, backend systems that perform the actions and Enterprise Service Buses (ESB) that serve as the middleware between CRMs and backend systems. This is why most companies are nowadays keen to use Operational Intelligence (OI) as a tool to gather all data into an understandable format.

Nevertheless, this opens a new problem with the arrival of the General Data Protection Regulation (GDPR). Companies are now facing a trade-off that can compromise the processing of personal data. Protecting data with standard encryption methods like the Advanced Encryption Standard (AES) leaves that same data in a non-processable format while not protecting it at all is risking the huge penalties that GDPR enforces. This not only forces these companies to abort the business of personal data processing and, with it, some of its most interested clients, as it also blocks any new solutions that were being prepared in the meantime.

This thesis studies a real-world scenario of a OI application whose intention is to process personal data in the future. This scenario fits in the description of the application whose new functionalities are currently on hold due to GDPR. It also uses a well-known OI platform named Splunk, which also suffers of the same trade-off problem.

The contribution of this thesis is to evaluate whether it is possible to protect data according to GDPR's requirements and still leave it in a format that is processable in the Splunk platform. For this, different homomorphic encryption algorithms will be tested and compared with the standard symmetric encryption algorithms on different big data scenarios.

Keywords

Operational Intelligence; General Data Protection Regulation; Homomorphic Encryption; Big data; Unstructured Data; Splunk; Service Oriented Architecture; Enterprise Service Bus;

Resumo

Cada vez mais vivemos num mundo que se move com o poder dos dados. Todo o tipo de aplicações, desde redes sociais a produtos negócio, produzem dados apenas por existir e realizar as ações que se esperam delas. Tome-se como exemplo uma empresa de telecomunicações onde são esperados milhares de clientes novos, quer residenciais quer de negócio. É esperado que, a partir do momento em que compramos algo nós estejamos também a fornecer informação sobre o que estamos a comprar e, acima de tudo, informação sobre o que somos. É do interesse das empresas que este tipo de informação seja mantida num formato que seja fácil de interpretar, de modo a inferir quais são os produtos que os clientes querem comprar e analisar o modo como essas compras são efetuadas para assegurar uma melhor manutenção do serviço. Estes sistemas são normalmente descentralizados o que significa que não é pouco comum a existência de sistemas secundários com propósitos bem definidos, como sistemas de gestão de relações com o cliente (CRM), sistemas de backend que realizam as operações que constituem o serviço e os chamados Enterprise Service Bus (ESB) que servem de *middleware* entre os CRMs e os sistemas de backend. É devido a esta complexidade de operações que as empresas dão cada vez mais valor ao uso de Inteligência Operacional como uma ferramenta para juntar dados num formato compreensível.

Porém, o uso destes dados abre caminho a um novo problema com a chegada da Regulação Geral de Proteção de Dados (RGPD). As empresas estão agora a enfrentar um *trade-off* que pode comprometer o processamento de dados pessoais. A sua proteção com métodos de criptografia *standard* como o *Advanced Encryption Standard* (AES) deixa os dados num formato não processável, enquanto não os proteger de todo significa arriscar o pagamento das multas impostas pela RGPD, o que não só força as empresas a abortar os seus negócios de processamento de dados como ainda bloqueia a implementação de soluções novas.

Esta tese estuda um cenário real de uma aplicação de Inteligência Operacional cuja intenção é a de processar dados pessoais num futuro próximo. Este cenário assenta na descrição da aplicação cujas novas funcionalidades estão em *stand-by* devido à RGPD. A aplicação usa uma plataforma de inteligência operacional chamada Splunk, que sofre também do mesmo problema do *trade-off.*

A contribuição desta tese é a de avaliar se é possível proteger dados em conformidade com os requisitos da RGPD e ainda assim mantê-los num formato processável pela plataforma Splunk. Para este efeito foram testados e comparados diferentes algoritmos de criptografia em diferentes cenários de *Big Data*.

Quatro soluções diferentes foram apresentadas: Anonimização, Criptografia Simétrica usando AES, Criptografia Homomorfica usando Paillier e Criptografia Homomorfica usando SEAL.

A solução de anonimização foi usada como modo de destruição de dados que é especialmente eficaz na concretização do direito ao esquecimento imposto pelo GDPR, embora comprometa os direitos de acesso, retificação e portabilidade dos dados da regulação. A anonimização tem também o revés de deixar os dados num formato não processável irreversível.

A solução de criptografia simétrica usando AES satisfez todos os requisitos da RGPD e ainda foi particularmente eficiente em termos de performance, segurança e complexidade, tendo apenas falhado a usabilidade dos dados após a sua proteção.

As soluções de criptografia homomorfica provaram ser eficientes quer em requisitos da RGPD quer em segurança, complexidade e usabilidade. No entanto, em termos de performance apenas a solução SEAL demonstrou potencial para ser aplicada numa aplicação de OI real.

Por fim, foram propostas duas soluções finais que usam os pontos mais benéficos de cada uma das soluções acima descritas para atingir objetivos específicos de RGPD, performance, segurança, complexidade e usabilidade. O único fator que distingue estas soluções é o momento em que estas protegem os dados pessoais: antes do armazenamento dos dados ou na leitura dos mesmos.

Palavras-Chave

Inteligência Operacional; Regulação Geral de Proteção de Dados; Criptografia Homomorfica; *Big data*; Dados não-estruturados; Splunk; Arquitetura Orientada a Serviços; *Enterprise Service Bus*;

Table of contents

Table index

Figure index

Acronyms, Abbreviations and Symbols

- GDPR – General Data Protection Regulation;
- OI – Operational Intelligence;
- EU – European Union;
- RoBAC – Role-Based Access Control;
- XML - eXtensible Markup Language;
- API – Application Programming Interface;
- DES - Data Encryption Standard;
- AES - Advanced Encryption Standard;
- RSA – Rivest Shamir Adleman (Ron Rivest, Adi Shamir, Leonard Adleman, designers of RSA);
- SHA – Secure Hash Algorithm;
- EAI - Enterprise Application Integration;
- ESB - Enterprise Service Bus;
- SOA - Service Oriented Architecture;
- CRM – Customer Relationship Management;
- ERP - Enterprise Resource Planning;
- B2B – Business-to-Business;
- OM – Order Management;
- IT – Information Technology;
- AM – Application Maintenance;
- SM – Service Manager;
- CM – Configuration Management;
- *sed* – Stream editor.

1 Introduction

In Big Data applications it's easy to mishandle and/or misuse critical data. This problem becomes even more important with the arrival of the General Data Protection Regulation (GDPR), which entered into force on the 25th of May of 2018 on all EU countries. This regulation was designed to harmonize data privacy laws across all Europe and its main objective is to empower all EU citizens data privacy.

This thesis focuses on a real-world Operational Intelligence solution that is implemented using the Splunk platform. The motivation behind this thesis is followed along with its objectives and contributions.

1.1 Motivation

More and more studies are being made to evaluate how well-prepared companies are for the arrival of GDPR, and basically all of them are concluding that business leaders are still confused about the regulation. Some of these studies go even further and say that roughly 90% of European companies are still not prepared and 40% have not started their preparations for the new regulations as of 20th of September of 2017 (Symcox, 2017). According to Veritas, 47% of organizations are expected not to meet the compliance guidelines (Stageberg, 2017).

As of today, the company whose project this thesis focuses on is avoiding personal data collection so that it isn't one of the aforementioned companies. Even though the processing of that same data brings good opportunities among their clients, the costs of not being complaint with GDPR are too immense to even consider such a possibility.

The project in question is an Operational Intelligence application implemented in Splunk whose purpose is to give insight into the ESBs of interested clients, by collecting and analysing its reliability, performance, volume and concurrency metrics. This application is currently being used by clients of the Telecommunications area where the processing of personal data opens good opportunities on the profiling of their customers. This makes it clear that the main gap on this project is that it doesn't have any way of profiling those customers, even though the customer information that arrives in the ESB of those companies is already sufficient for that matter.

Splunk does already offer the means to protect personal data in order to be compliant with GDPR, nevertheless, none of them offers means of processing that data without decrypting it first (Splunk, 2017). As suggested by Splunk, one could protect the data before it is stored but, with the current standard encryption methods that are considered by Splunk (eg.: AES), the data would be on a format that can no longer be processed, which gets us back to the original problem of not processing the data at all or risking the penalties that are enforced by GDPR.

Nevertheless, a new kind of encryption seems to be increasing in popularity to face this problem. Homomorphic encryption offers the possibility to process data even after it is already encrypted, which seems to be a good way to go.

1.2 Objectives

The objective of this thesis is to build a solution that is both compliant with GDPR and processable by Splunk. This should be done in a manner that the non-functional requirements like performance, security, usability and complexity of the OI application are compromised as little as possible.

In order to achieve this, the following secondary objectives need to be accounted on this thesis' development:

- Establish a simulation scenario that can be used to evaluate this thesis;
- Identify possible encryption solutions, be they homomorphic or not, for different Operational Intelligence scenarios;
- Compare those encryption solutions and identify pros and cons for each one of them;
- Understand if these solutions are compliant with the General Data Protection Regulation.

1.3 Contribution

This thesis contributes with a data protection requirement analysis that is compliant with GDPR, while also establishing which data can be processed, which should be protected and which should not be processed at all. Non-functional requirements like performance, security, usability and complexity are also be established.

For this matter, different Splunk architectures were tested along with both symmetric and homomorphic encryption algorithms, thus allowing a direct comparison between multiple prototypes on the aforementioned requirements.

Based on those prototypes, a few solution proposals were made in the end of this report with different trade-offs in the requirements that they comply with.

1.4 Document structure

First, all background work that was found to support the objectives of this thesis is introduced in Section 2. This section highlights the most important articles of the General Data Protection Regulation, some encryption techniques that should be considered and the main concepts that drive the studied OI application, namely SOA, ESB, Operational Intelligence and Splunk.

Section 3 follows with the analysis and design of both the objectives and contribution of this thesis. It makes an introduction of the studied OI application by explaining what are its involved subjects, its use cases, operations and types of data, which lead up to the scenarios and requirements that should be considered on the "04 Prototypes" and "05 Proposed Solutions" sections.

Section 4 starts by introducing the business scenario of the OI solution and, based on that, analyses four possible prototypes, along with the technical decisions that were made. Having in mind these prototypes, two final solution proposals are made using the best qualities of each.

Finally, section 5 builds a conclusion upon all the work that was done on this thesis. The document ends with a full list of references that were made on the text of this document.

2 Background work

This thesis addresses the General Data Protection Regulation, along with its main concepts and articles, addresses the most well-known encryption techniques and uses Splunk as its operational intelligence platform to give insight on data of SOA driven ESBs.

2.1 General Data Protection Regulation (GDPR)

The GDPR lays down rules regarding the protection of natural persons' personal data and the movement of that data within the EU. This regulation applies to any organization that collects or processes any personal data of EU residents, be they inside or outside the Union, which means that most of the worldwide organizations will have to deal with this regulation at some point.

2.1.1 Scope

In regard to operational intelligence applications, the following key concepts are defined on the GDPR:

Personal data:

"'*personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person*". (General Data Protection Regulation - Final text neatly arranged, n.d.)

Processing:

"'processing' means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction". (General Data Protection Regulation - Final text neatly arranged, n.d.)

Controller:

"'controller' means the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law;" (General Data Protection Regulation - Final text neatly arranged, n.d.)

Processor:

"'processor' means a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller;" (General Data Protection Regulation - Final text neatly arranged, n.d.)

2.1.2 Principles

In regard to operational intelligence applications, the following principles from GDPR should be considered.

a. Processing of personal data (art. 5)

Personal data should be processed according to the following principles:

- **Purpose limitation** - "*Collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes*"
- **Data minimisation** - "*Adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed*".
- **Accuracy** - "*Accurate and, where necessary, kept up to date*", meaning that, when inaccurate, the data should be erased or rectified without delay.
- **Storage limitation** - "*Kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed*".
- **Integrity and confidentiality** - "*Processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures*".

In summary, the controller/processor should have as less personal data as possible, should have accurate data, should get rid of the data when it is no longer needed and should protect the data while it is being used/stored.

b. Processing of special categories of personal data (art. 9)

Processing is prohibited for the following types of personal data:

- Racial or ethnic origin;
- Political opinions;
- Religious/philosophical beliefs;
- Trade union membership;
- Genetic data;
- Biometric data;
- Health;
- Sex life;
- Sexual orientation.

The above types of personal data can still be processed if:

- "Data subject gives **explicit consent**";
- "Processing is necessary for carrying out obligations and exercising specific rights of the controller or of the data subject in the field of employment and social security and social protection law";
- "Processing relates to personal data which are **manifestly made public** by the data subject";
- "Processing is necessary for the **establishment, exercise or defence of legal claims** or whenever courts are acting in their judicial capacity";
- **"**Processing is necessary for archiving purposes in the public interest, scientific or historical research purposes or **statistical purposes";**

For operational intelligence purposes, these types of data should be protected right away and can only be used for statistical purposes. Still, the controller/processor should be able to revert the protection using appropriate methods, in case it faces a legal claim.

c. Processing which does not require identification (art. 11)

According to the regulation, the controller is not obliged to maintain, acquire or process additional information to identify the data subject if the data in question does not require the data subject's identification.

This will be essential for data bulks that have no personal data, for which companies can simply use the data without any preoccupation.

2.1.3 Rights of the data subject

According to GDPR, data subjects have the following rights regarding their personal data.

a. Right of access by the data subject (art. 15)

The data subject has the right to obtain confirmation as to which personal data concerning him is being processed and what are its purposes. The following information can be obtained:

- Purposes of the processing;
- Categories of the personal data;
- Recipients or categories of recipient to whom the data is or can be disclosed;
- Envisaged period for which the personal data will be stored;
- Copy of the personal data currently undergoing processing.

b. Right to rectification (art. 16)

The data subject has the right to have innacurate personal data that concerns him rectified without undue delay.

c. Right to be forgotten (art. 17)

The data subject has the right to erasure of all personal data concearning him or her without undue delay.

d. Right to data portability (art. 20)

The data subject has the right to receive all the personal data concerning him or her in a well-enough structured format that allows that same data to be transmitted to another controller.

2.1.4 Controller and processor

The **controller** is, according to the GDPR, responsible to determine the purposes and means of the processing of personal data. On the other hand the **processor** is someone or some entity that works for the controller, processing personal data on his behalf.

This section shows some important articles regarding controllers and processors.

a. Data protection by default and by design (art. 25)

When personal data is not needed for specific purposes, it should be protected using appropriate technical and organisational measures. The regulation refers to **pseudonymisation** as an example of one of these measures.

These measures should take into account the amount of data, extent of their processing, the period of their storage and their accessibility.

b. Records of processing activities (art. 30)

According to the regulation, the controller should maintain a record of its processing activities. This record should contain the following information:

- Name and contact details of the controller;
- Purposes of the processing;
- Description of the categories of data subjects and their personal data;
- Categories of recipients to whom the personal data have been or will be disclosed;
- Envisaged time limits for erasure, if possible;
- General description of the technical and organisational security measures.

c. Security of processing (art. 32)

The controller and the processor need to ensure a level of security appropriate to the risk, which includes:

- Pseudonymisation and encryption of personal data;
- Ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services;

2.2 Access control

Access control is a security technique that is used to specify what users can do, which resources they can access and what operations they can perform on those resources in a computing environment. For the purposes of this thesis, only the Role Based Access Control will be used, which already comes as part of the Splunk platform.

2.2.1 Role-Based Access Control (RoBAC)

The role-based access control policy assigns permissions to particular roles on an organization. Users are then assigned to a role as appropriate.

2.3 Encryption

For the purposes of this thesis, three types of encryption will be considered:

- Symmetric Encryption
- Asymmetric encryption
- Homomorphic encryption

Sections 2.3.1 and 2.3.2 were based on the book “Computer Security: Principles and Practice (Stallings & Brown, 2015), while section 2.3.3 was based on the book “Homomorphic Encryption and Applications” (Yi, Paulet, & Bertino, 2014).

2.3.1 Symmetric encryption

As the name suggests, this type of encryption algorithm uses the same key to both encrypt and decrypt data, which makes the key symmetric in terms of usage. In practical terms, this key represents a secret

that is shared between two interested entities. Figure 2.1 illustrates the course of action of this type of encryption.

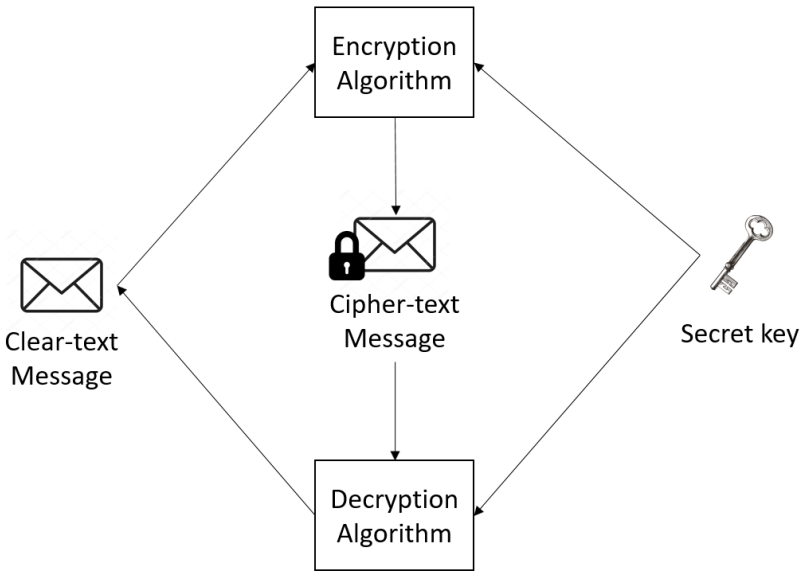

Figure 2.1 - Symmetric encryption scheme

There are two types of Symmetric Ciphers:

- **Stream ciphers -** ciphers that process input elements in a continuous fashion, producing output for each element individually.
- **Block ciphers –** ciphers that process the input in blocks of elements, producing an output for each block individually.

a. DES

The Data Encryption Standard (DES) was implemented in 1977 by the American government with the objective of encrypting/decrypting non-classified data. This algorithm has been criticized ever since due to the low size of the keys it uses, for having that are not clear in terms of purpose and for being easily broken by a lot of already known algorithms.

This algorithm uses a block length of 64 bits and keys of 56 bits. It is a minor variation of the Feistel network, which is illustrated on Figure 2.2, with 16 rounds and 16 subkeys that are generated from the original 56 bit key.

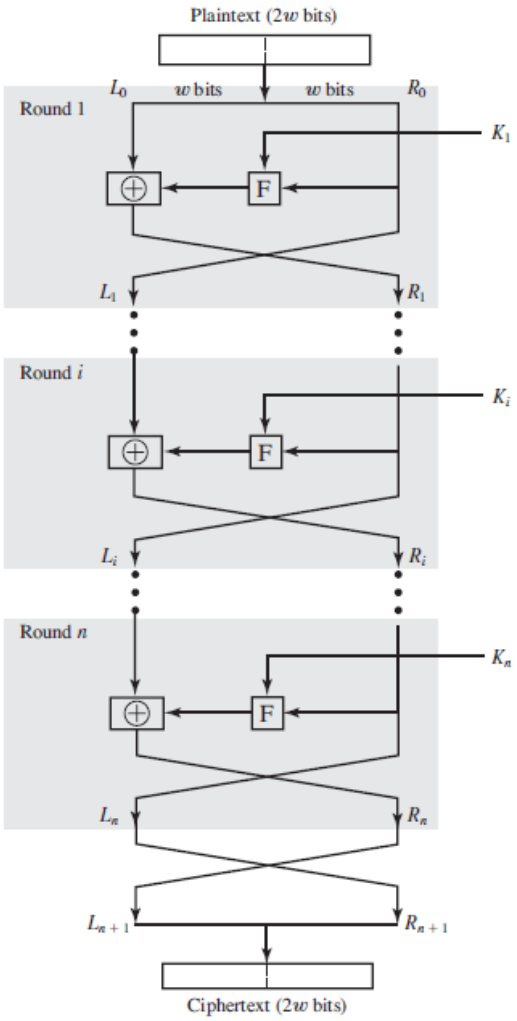

Figure 2.2 - Feistel Network

In the Feistel network we start by having a plaintext block with *2w* bits of length and a key *K*. Then, the alogorithm performs the following steps:

1. The plaintext is divided in two halves, L0 and R0, of equal size *w*.
2. A subkey *Ki* is generated from the original key K with a subskey generation algorithm.
3. The right half of the data is used on the round funtion *F* along with a subkey *Ki*.
4. A substitution is performed on the left half of the data by taking the *exclusive-OR* (*XOR*) of itself and the output of the function *F*.
5. The result of the substitution on the left side goes to the right side while the right side goes to the left side unchanged.
6. Repeat for N rounds.
7. Merge the resulting left and right halves into a final ciphertext with *2w* bits.

b. Triple DES

Triple DES (3DES) was standardized with the objective of being used on financial applications in 1985, being also incorporated as part of the Data encryption Standard in 1999.

This algorithm uses three keys and three executions of the DES algorithm, following an encrypt-decrypt-encrypt sequence, as shown on Figure 2.3.

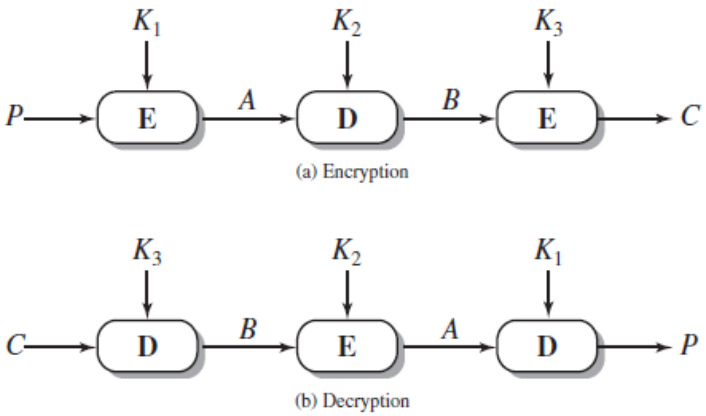

Figure 2.3 - Triple DES scheme

c. AES

The Advanced Encryption Standard was implemented with the purpose of being the federal information processing standard and replacing the publicly considered weak counterparts DES and Triple DES.

This algorithm uses a block length of 128 bits and a key length that can be 128, 192 or 256 bits. It is composed by one permutation stage and three substitution stages:

1. Substitute bytes – Uses a table named "*S-box*" to perform a byte-by-byte substitution of the block;
2. Shift rows – A simple permutation that is performed row-by-row;
3. Mix columns – A substitution that alters each byte in a column as a function of all its bytes.;
4. Add round key – A simple bitwise XOR of the current block with a portion of the expanded key.

These stages are then repeated for 10 rounds, before reaching the intended cypher-text. Figure 2.4 illustrates the whole process.

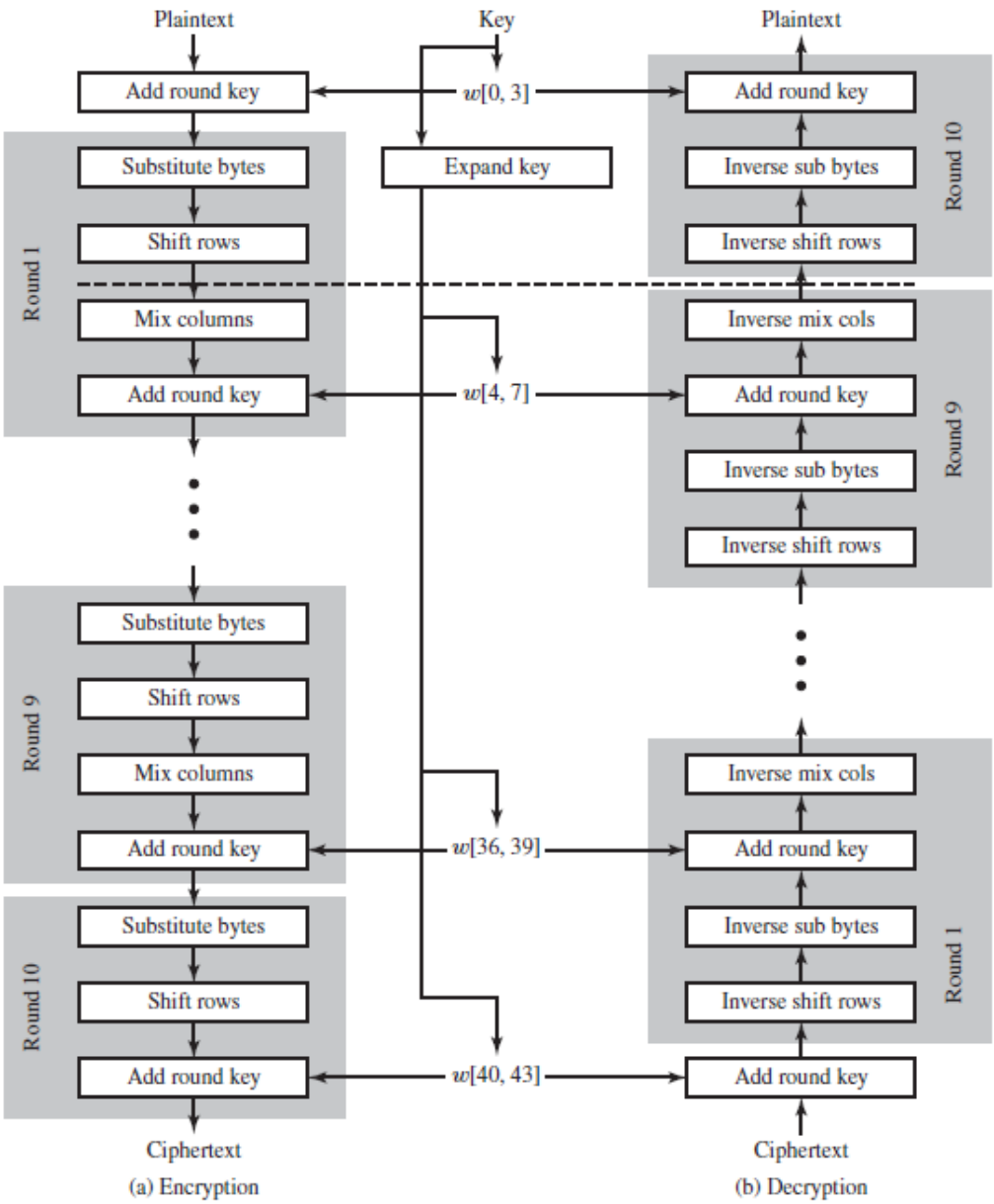

Figure 2.4 - AES scheme

2.3.2 Asymmetric encryption

Also known as Public-Key Encryption, Asymmetric Encryption is, as the name suggests, the algorithm that uses keys that are asymmetric, meaning that they can only be used for either encryption or decryption, never both at the same time.

Two keys are needed in this algorithm, one which is known as the public key and the other which is the private key. The public key is known by all the involved entities while the private key is supposed to be known only by the entity to which it belongs to. This also means that different results can be achieved when we use one or the other for the encryption:

1. Encrypting with the public key of entity X forces the decryption to be done with the private key of entity X, which is a way of guaranteeing that only entity X is able to read the date in clear-text-

2. Encrypting with the private key of entity X can only be done on the entity X's premises and is a way for entity X to prove its identity.

a. RSA

This algorithm was developed in 1977 and remains to this day as the most widely accepted approach to public-key encryption. RSA is a block cipher in which plaintext and ciphertext are integers between 0 and n-1 for some n. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$C = M^e mod\ n$

$M = C^d\ mod\ n = (M^e)^d\ mod\ n = M^{ed}\ mod\ n$

It is assumed that both sender and receiver must know the values of *n* and *e*, and only the receiver knows the value of *d*. The public and private keys are here represented on the following form:

Public key: $PU = \{e,n\}$

Private key: $PR = \{d,n\}$

This algorithm is only satisfactory when the following requirements are met:

1. It is possible to find values of *e, d, n* such that $M^{ed}\ mod\ n = M$, for all $M < n$.
2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is infeasible to determine *d* given *e* and *n*.

2.3.3 Homomorphic encryption

Homomorphic encryption is the category of encryption that allows computation to be performed on encrypted data. The purpose is to have exactly the same result after decrypting the data as if the computation was done on plaintext.

Homomorphic encryption can be either **partially homomorphic** or **fully homomorphic.**

a. Fully homomorphic schemes

These homomorphic encryption schemes allow any kind of computation to be performed on the encrypted data, without any limitation. This is by far the most powerful scheme since it allows for any desirable functionality to be implemented.

In practical terms, homomorphic encryption schemes have no clear solution yet, since it wasn't yet found a formula that can be applied without doubt to any kind of data, be it numeric or alpha-numeric.

b. Partially homomorphic schemes

This homomorphic encryption scheme only allows a subset of operations to be performed on the encrypted data while ensuring that the decryption returns the expected result.

These schemes are usually based on conventional cryptography and, as such, share the same performance and security levels.

Unpadded RSA

If the RSA public key is modulus *m* and exponent *e*, then the encryption of a message *x* is given by: $\mathcal{E}(x) = x^e \bmod m$. The homomorphic property is then:

$\mathcal{E}(x1).\ \mathcal{E}(x2) = x_1{}^e x_2{}^e \bmod m = (x_1 x_2)^e \bmod m = \mathcal{E}(x1.x2)$

ElGamal

In the ElGamal cryptosystem, in a cyclic group G of order *q* with generator *g*, if the public key is (*G*,*q*,*g*,*h*), where $h = g^x$, and *x* is the secret key, then the encryption of a message *m* is $\mathcal{E}(m) = (g^r, \text{m}.\ h^r)$, for some random $r \in \{0,...,\text{q-1}\}$. The homomorphic property is then:

$\mathcal{E}(m1).\mathcal{E}(m2) = (g^{r_1}, m1.\ h^{r_1})\ (g^{r_2}, m2.\ h^{r_2}) = (g^{r_1+r_2}, (m1.m2).\ h^{r_1+r_2}) = \mathcal{E}(m1.m2)$

Goldwasser-Micali

In the Golwasser-Micali cryptosystem, if the public key is the modulus *m* and quadratic non-residue *x*, then the encryption of a bit *b* is $\mathcal{E}(b) = x^b r^2 \bmod m$, for some random $r \in \{0,...,m-1\}$. The homomorphic property is then:

$\mathcal{E}(m1).\mathcal{E}(m2) = x^{b_1}\ r_1{}^2 x^{b_2} r_2{}^2 \bmod m = x^{b_1+b_2}\ (r_1\ r_2)^2 \bmod m = \mathcal{E}(b1 \oplus b2)$

Benaloh

In the Benaloh cryptosystem, if the public key is the modulus *m* and the base *g* with blocksize of *c*, then the encryption of a message *x* is $\mathcal{E}(x) = g^x r^c \bmod m$, for some random $r \in \{0,...,m-1\}$. The homomorphic property is then:

$\mathcal{E}(m1).\mathcal{E}(m2) \bmod m = (g^{x_1}\ r_1{}^c)\ (g^{x_2}\ r_2{}^c) \bmod m = g^{x_1+x_2}\ (r_1\ r_2)^c = \mathcal{E}(x_1 + x_2)$

Paillier

In the Paillier cryptosystem, if the public key is the modulus *m* and the base *g*, the encryption of a message *x* is $\mathcal{E}(x) = g^x r^m \bmod m^2$, for some random $r \in \{0,...,m-1\}$. The homomorphic property is then:

$\mathcal{E}(x1).\mathcal{E}(x2) = (g^{x_1}\ r_1{}^m)\ (g^{x_2}\ r_2{}^m) \bmod m^2 = g^{x_1+x_2}\ (r_1 r_2)^m \bmod m^2 = \mathcal{E}(x1 + x2)$

2.4 Operational Intelligence Application

Operational Intelligence is a category of real-time business analytics whose main objective is to give insight into data, be it on the form of streaming events or business operations. This data insight allows users to take conclusions in regard to the system's performance, threats and process workflow while correlating them together to reveal patterns, detect events, identify anomalies and come up with a solution.

The Operational Intelligence application in study has 4 main components: **Probe**, **Forwarders**, **Indexers** and **Search Heads**.

Probe is the component that collects the data from the ESB that the OI application monitors.

Forwarders are the components that consume the data and forward it to Indexers. They reside on the machines from where the data originates and do little or no work on the data, forwarding it in a raw format.

Indexers are the components that process the incoming machine data, converting them from unstructured data to structured data. This data is stored on indexes as events, which can later on be interpreted in different ways by search heads.

Search heads are the components that allow users to search the already indexed data. The results of these searches can be returned in the form of field value pairs or consolidated/processed in some way, according to the user's needs.

Figure 2.5 illustrates of the OI solution showcasing the aforementioned components (in green), the ESB that is being monitored and the surrounding environment.

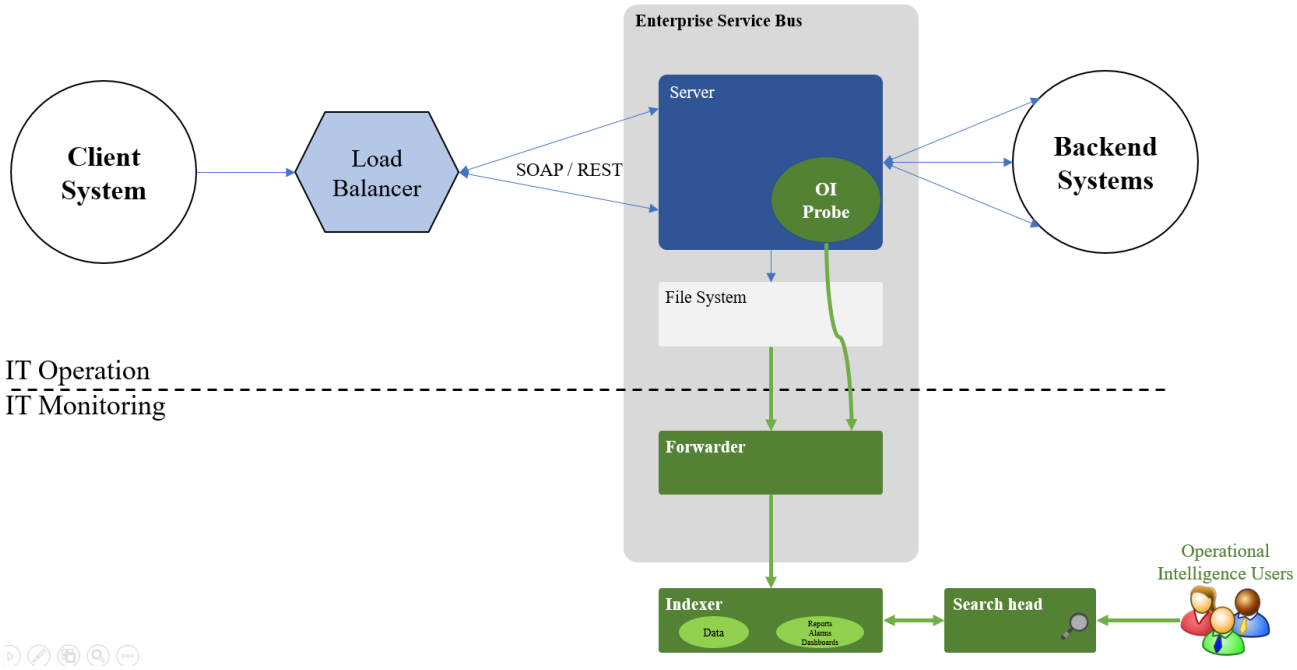

Figure 2.5 - OI solution architecture diagram

In summary, a probe was installed on the server on which the client application resides, so that all the data that is needed to monitor the environment can be obtained to perform the OI studies. This data is then sent by this probe to a forwarder server that resides on that same machine.

The forwarder sends all the data in raw format to the indexer servers, where it is indexed thus becoming structured.

Finally, the OI users login in to the OI application on the Search Head and use it to consult the events that were collected throughout this process.

2.5 Splunk

The Operational Intelligence solutions that are analysed on this thesis are all implemented in Splunk. Splunk uses the same concepts as shown on section 2.4 regarding the operational intelligence application, meaning that typically it has:

- A target system where a probe should reside;
- Forwarders;
- Indexers;

- Search heads.

Each one of these components handles data that may include personal information. All the concepts that will be introduced on the following sections were taken from Splunk's documentation pages (Splunk, 2017).

2.5.1 Splunk data pipeline

Splunk has 3 tiers to handle data:

Tier	Splunk server that is responsible for this tier
Data input	Forwarder
Indexing	Indexer
Search management	Search Head

Table 2.1 - Splunk data pipeline tiers

Figure 2.6 illustrates the route that data takes through these tiers, which is called data pipeline.

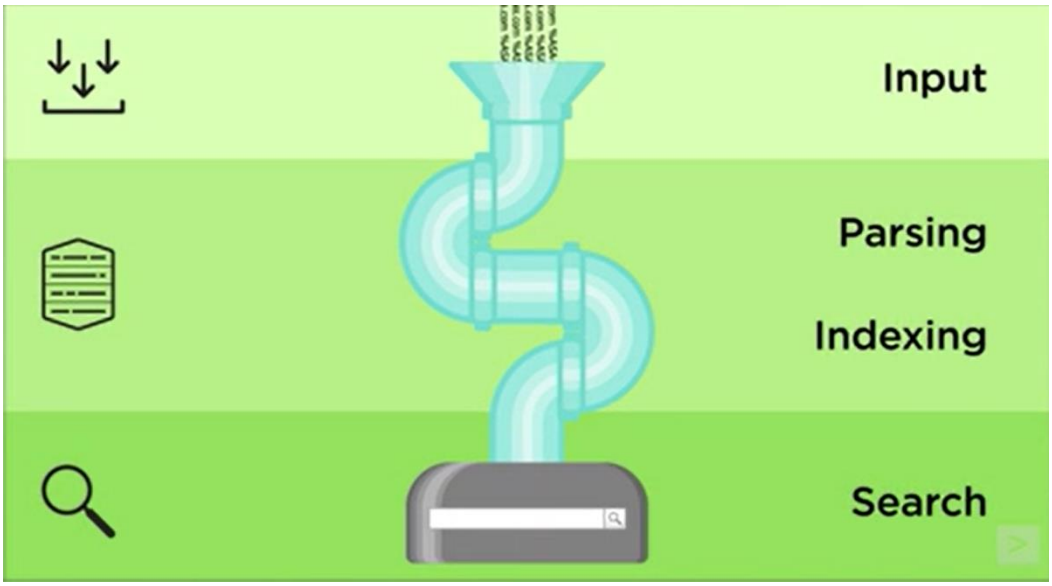

Figure 2.6 - Splunk data pipeline

On the **input** tier, Splunk acquires raw data from the source and breaks it into equally sized blocks (64K). Metadata like the source type, the specific source and the host is also obtained on this tier. Events are still non-existent on this phase, meaning that the data is still in its original format.

The **indexing** tier is composed by a parsing phase and an indexing phase. During the **parsing** phase, Splunk does the following:

- Breaks the data streams into single lines;
- Identifies, parses and sets timestamps;
- Adds source-wide keys that were created at input time;
- Transforms event data and metadata based on regular expressions.

During the **indexing** phase, Splunk writes events on the indexer's disk and indexes:

- Compressed raw data files;
- Corresponding index files.

The **search** tier defines how each user accesses, views and uses indexed data. This is basically where users create the analytics that'll be shown to the end client.

2.5.2 Conversion to structured data

The conversion of incoming unstructured data is done during the indexing tier and is directly related with the way Splunk processes incoming data and stores it in indexes, which are basically repositories for data. As was explained on section 2.5.1, indexing is where raw data is converted into searchable events.

It's during the **parsing** phase of the indexing tier that data manipulation is done. Here, Splunk:

- Extracts a set of default fields for each event, including host, source, and sourcetype;
- Configures character set encoding;
- Identifies line termination using line-breaking rules. While many events are short and only take up a line or two, others can be long;
- Identifies timestamps or creates them if they don't exist. At the same time that it processes timestamps, Splunk identifies event boundaries;
- **Masks sensitive event data**, if the appropriate configuration was done for that. It can also be configured to apply custom metadata to incoming events.

The masking of sensitive data is the key action here and it's essential to know how Splunk is able to identify this type of data. Splunk has 2 ways of anonymizing data:

- With regular expression (regex) transformations;
- With "*sed*" scripts.

a. Regex transformations

This type of anonymization requires the user to configure two Splunk configuration files:

- props.conf – where the wanted transformation classes are specified for specific sourcetypes, hosts or sources;
- transforms.conf – where the actual transformation classes are implemented with the regex that matches the data to anonymize.

b. *sed* Script

This type of anonymization requires the user to create a sed script. Sed is a Unix utility that reads a file and modifies the input as specified by a list of commands. The script needs to be configured first on the props.conf file in a similar way as in regex transformations.

Splunk allows 2 types of *sed* commands:

- replace – Uses regex to find an expression and replaces that expression by a given string.
- character substitution – Searches for a specific set of characters and replaces them by a set of given characters.

2.5.3 Data obfuscation and Field Protection

As was presented on Splunk's ".conf17" (Splunk, 2017), we have several options when implementing a solution that protects data on Splunk. This section explains those options having in mind that the data protection on the transport level is already assured by Splunk with safe channels (using TLS).

a. Application

On this option, it is the data source's responsibility to protect the data before sending it to the Splunk system. This implies that little to no latency is introduced to the system and security is very high but at the same time is a bit complex since this implementation needs to be prepared for each client that will use the Splunk implemented solution (different client, different data). Also, the usability is not the best since this implies that the client needs to protect all data that is thrown to Splunk, even data that was just now introduced, which most likely won't happen.

Option	Layer	Latency	Security	Complexity	Usability
Application	Data source	Low	Very High	Medium	Medium

Table 2.2 – Splunk's data protection options - Application

b. Modular input / Batch processing

This option protects the data on the forwarder with encryption, pseudonymization and anonymization. This offers high usability and security since it allows specific operations to be performed on certain events or types of data which can protect the data while leaving it still usable (eg.: Format preserving encryption). Nevertheless, it is complex to implement such a solution and it induces delay due to the API calls.

Option	Layer	Latency	Security	Complexity	Usability
Modular Input / Batch processing	Data	Medium	High	High	High

Table 2.3 - Splunk's data protection options - Modular input/Batch processing

c. External Processor

This option requires an external processor, which in itself is an additional solution that should be implemented from scratch, updated each time a new client starts using the Splunk solution an maintained. Obviously, this is the solution with the highest complexity and has still medium latency due to the API calls to the processor. Nevertheless, this solution offers also important advantages in terms of usability and security.

In terms of architecture, the external processor receives the data that needs to be protected from the forwarder and then send it to the indexer.

Option	Layer	Latency	Security	Complexity	Usability
External Processor	Data	Medium	High	High	High

Table 2.4 - Splunk's data protection options - External Processor

d. Regex Replace

This option protects the data on the indexer by replacing it with some other value each time a regular expression is matched. This has a high security and low complexity as the main positive aspects since it anonymizes the data and is quite simple to configure. Nevertheless, it adds latency to the system and is not usable on search time, due to that same anonymization.

Option	Layer	Latency	Security	Complexity	Usability
Regex Replace	Data	Medium	High	Low	Low

Table 2.5 - Splunk's data protection options - Regex Replace

e. Scheduled Search

This solution protects the data with scheduled searches that are configured to find sensitive data from time to time and run a script that protects them. Of all the options, this is the most expensive one in terms of latency, obviously. It also offers poor security, because between different scheduled searches, the data will remain unprotected. Finally, the complexity and usability of this solution is medium because the searches that find sensitive data are not straightforward and the latency affects the usability.

Option	Layer	Latency	Security	Complexity	Usability
Scheduled Search	Data	Very High	Low	Medium	Medium

Table 2.6 - Splunk's data protection options - Scheduled Search

f. Result Masking

This option is implemented on the presentation layer instead of the data layer as in the options above. This is the most common solutions since it is "good enough" for approximately 90% of the scenarios, since it offers really high usability and really low complexity. Nevertheless, if offers low security because the data is stored in clear on all components of Splunk and offers a bit of latency as well on the presentation of data, since the protection is done there.

Data can be protected with 3 different methods in Splunk:

- Anonymization – uses SHA256 hash.
- Pseudonymization - uses AES256 encryption.
- Format preserving Pseudonymization - Based on AES as well but preserves the original format.

Option	Layer	Latency	Security	Complexity	Usability
Result Masking	Presentation	Medium	Low	Low	High

Table 2.7 - Splunk's data protection options - Result Masking

g. Options Summary

In the realm of Operational Intelligence, the most desirable solution would be to have as much Usability as possible while having a high level of Security. Table 2.7 already shows that no such solution exists at the moment. In order to have good security, one needs to compromise on Usability, and vice-versa.

Option	Layer	Latency	Security	Complexity	Usability

Application	Data source	Low	Very High	Medium	Medium
Modular Input / Batch processing	Data	Medium	High	High	High
External Processor	Data	Medium	High	High	High
Regex Replace	Data	Medium	High	Low	Low
Scheduled Search	Data	Very High	Low	Medium	Medium
Result Masking	Presentation	Medium	Low	Low	High

Table 2.8 - All Splunk data protection options

2.5.4 Access control

Splunk already comes with a built-in Role Based Access Control policy that can be applied at search time. This policy has 3 types of users:

Role	**Description**
User	Intended for the common user that edits its own searches, knowledge objects. This user can only see its own objects or objects that are specifically shared to him.
Power-User	Intended for users that can edit all shared objects, alerts, tag events and other similar tasks.
Administrator	Intended for administrators that are able to manage most of the system's users. Has the most assigned capabilities (egs.: edit_roles, edit_user, edit_server,…)

Table 2.9 - Splunk's access control roles

New roles with different permissions can be added if needed.

2.6 SOA

Service Oriented Architecture (SOA) is, as the name suggests, a software architecture where all functionalities that are implemented by the applications are available in the form of services. This architecture follows a consumer-provider philosophy, meaning that all functionalities are provided by some entity to another that consumes them.

SOA defines a service as a unit of work representing a specific business function. This unit of work should be accessible through well-defined interfaces. Services are normally implemented using the Simple Object Access Protocol (SOAP) that provides a specification for message exchanging, and Representational State Transfer (REST) which is a set of principles that define how Web Standards should be used (Hurwitz, Bloor, Kaufman, & Halper, 2009).

2.6.1 SOA Infrastructure

SOA service infrastructure can be achieved with Service Enablement and Service Mediation.

a. Service Enablement

Service Enablement relates to the service implementation part we use to enable services. If we want to adopt SOA practices, the first thing we have to do is search for some technology that already implements the service we want. This technology may already exist in our company, it may as well already work the way we want and by doing this we're accelerating the process and saving funds for future service implementations. Service enablement can be achieved with an *Enterprise Service Bus (ESB),* which is also a mediator and works well with applications with good interfaces and can be achieved with an *Application Wrapper* that provides a way of using services that don't have a formal interface by exposing running applications or internal programs as services.

b. Service Mediation

Service Mediation relates to the separation between service consumers and service providers. A service consumer should never connect directly to the service provider. If it does so, there's absolutely no flexibility in our SOA practice regarding the changes we can make to our services. This is of extreme importance because services are constantly changing! New features, new requirements, corrections, new message formats, new programming languages, anything we can think of will most likely change in the future. To achieve service mediation, we have to implement virtual service that functions as a translator between services. By doing so, we achieve **Loose coupling** which is the property where service consumers are independent of the service provider's location, transport and type of messages. This can be made with Proxies, ESB's, Gateways and SOA appliances.

2.6.2 Governance

The only way to fight the aforementioned problems is by defining roles, policies and procedures and by enforcing those to all stakeholders. This is called *Governance*. The enforcement of this roles, policies and procedures can be made in *design-time* and *run-time* with ***checkpoints***.

Enforcement in design-time is made with **registries**, where we publish the available services and related metadata, SOA related assets, service provider organizations, service consumer systems or applications, service consumer organizations, policies, contracts, agreements and relations between all. What makes this method a checkpoint of policies is that all system artefacts must pass through the registry before being available to service consumers.

Enforcement in run-time is made with **Policy Enforcement Points (PEP)**, also known as **Brokers**. PEP's provide a way of enforcing pre-established agreements between service providers and service consumers. PEP's are the service-mediation layer of the SOA infrastructure, which we will see next. By enforcing policies in the mediation layer, we can apply policies without worrying about both service consumer and service provider locations and protocols and message differences.

a. SOA Life Cycle

Governance shouldn't be a burden to workers. In fact it should always be seen as a way of helping the workers do their job. What usually happens is that workers publish a certain service on the registry, wait for the checkpoint and then have to restart all their work again because the service wasn't compliant with the company policies.

The main objective of life cycles is to fix the above issue by breaking the development process of a service into stages. In each stage, some person, that is responsible for some task, evaluates the work

according to the policies of the company. The normal service life cycle has ***requests*** made by a *Business Owner*, which are analysed and turned into service and process **designs** by an *SOA Architect.* These designs are then implemented by a *developer* on the **develop** stage who also makes **tests** for the same implementations. These tests are used by the *Quality Manager* to validate quality metrics, side-effects and non-functional requirements such as quality of service. The *Operator* receives the well-tested and validated solutions and starts the **production** stage, where he makes the solutions available for service consumers by implementing a virtual service.

All the aforementioned stages of the SOA life cycle are illustrated on Figure 2.7.

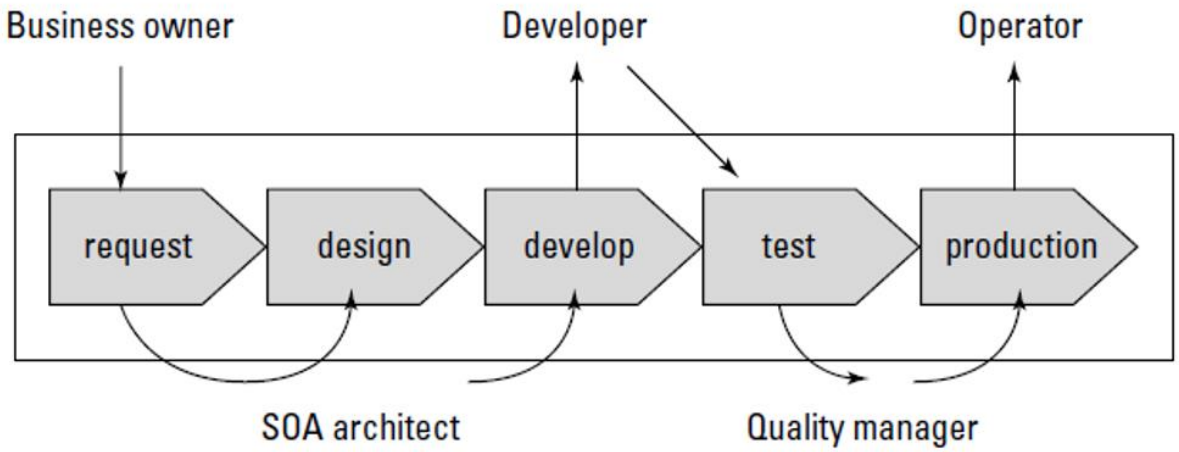

Figure 2.7 - SOA Life Cycle

b. Composition

SOA life cycles also imply that the stakeholders mentioned above can communicate with each other. This originates a new problem in that, most of the time, those same stakeholders have different languages and skills. A *Business Owner* may not know how to design services, an *SOA Architect* may not know what to implement a design, and so on.

The commonly used solution is **Business Process Management (BPM)**. BPM provides a way of composing and optimizing the SOA process by defining processes in a very graphical fashion, which results in a common language to all stakeholders. Figure 2.8 shows an example of a BPM model.

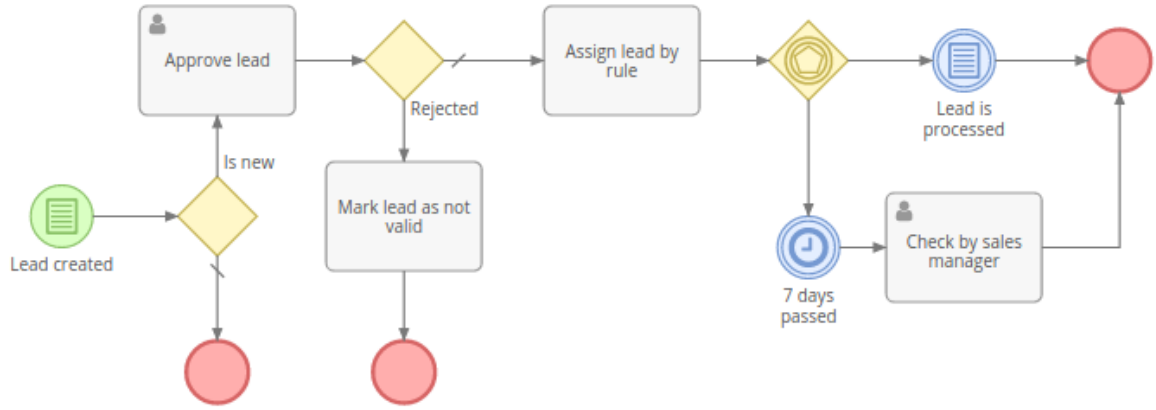

Figure 2.8 - Example of a Business Process Management model

Another way of composing the SOA process is by developing *Composite Applications*. These technologies provide a new way of creating applications where all that is required is wiring up services into a rich user interface. This makes reuse of services accessible to everyone, since little or no code is required.

c. Agility

Composition and life cycles brings us to one big objective of SOA: **Agility**. Agility means the capability to rapidly and cost efficiently adapt to changes. To achieve agility, we have to ensure that all stakeholders use shared technical and organizational components and functions as well as good relationships to increase communication and reach a common goal. We need to assure a ***mindset*** where everyone understands that the service exists to deliver consumer satisfaction, ***methodology*** to cross project boundaries and improve communication, and ***challenges*** to the organization by defusing any conflicts between departments. Another way of increasing agility is by defining contracts. Contracts define an agreement between the non-functional requirements of consumers and providers, like performance, security, and so on.

2.7 ESB

The real-world OI solution that will be protected in the scope of this thesis gives insight into the execution of an Enterprise Service Bus (ESB).

An ESB is a form of message-oriented middleware (MOM) system that implements the communication between software applications in a service oriented architecture (SOA) (Chappell, 2009). As illustrated on Figure 2.9, the primary responsibilities of an ESB are the following:

- Routing of messages;
- Monitoring and control message exchange;
- Event handling;
- Data transformation and mapping;
- Message and event queueing/sequencing;
- Protocol conversion;
- Exception handling;

- Security.

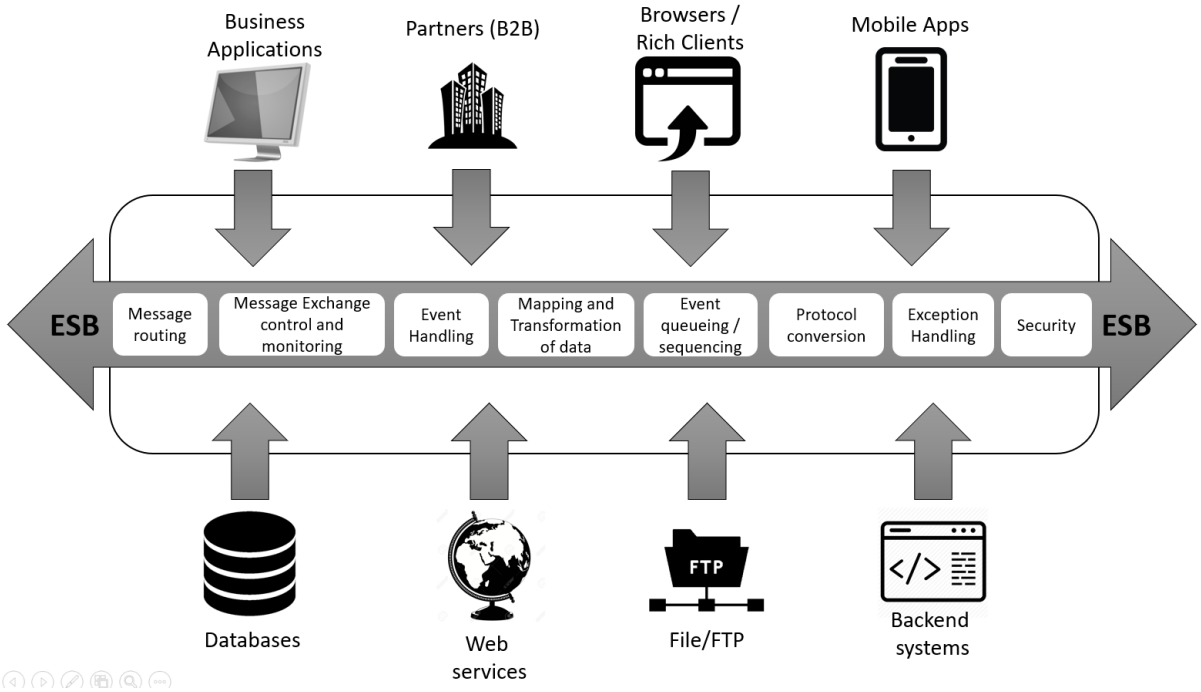

Figure 2.9 - Enterprise Service Bus

2.8 Summary

This section addressed GDPR, along with its main concepts and articles, encryption techniques, Splunk Operational Intelligence, SOA and ESB, which are the concepts that drive this thesis.

GDPR focuses on the protection of personal data, empowering common individuals over their own data. This regulation specifies how this type of data should be handled/processed and how it should be protected, making it the main source of requirements for this thesis.

On the other hand, the access control sub-section makes a small introduction into the Role-Based Access Control policy, which is the one that is used by Splunk.

Symmetric, asymmetric and homomorphic encryption algorithms are introduced here as well as a way of protecting data. Both symmetric and asymmetric encryption schemes represent the existing solutions that allow the protection of sensitive data without regard to the operations that were being executed on it. On the other hand, homomorphic encryption schemes represent the possibility of implementing operations that can be executed on encrypted data, to allow users without access to that kind of data to still do some amount of work in the scope of an organisation.

This section also makes a quick introduction to the Operational Intelligence field of work, following immediately with its main platform, Splunk. Since this thesis focuses on the security side of data processing, themes like the different stages that data goes through on this platform are introduced. The different data protection options that are already offered by Splunk for each individual stage were introduced here along with their pros and cons.

Finally, an introduction was made to the Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB) concepts, which are tightly related with the OI application that will be improved in the scope of this thesis.

Section 3 introduces the model of current existing OI solution, by explaining its most important concepts, its architecture, the involved subjects, use cases, operations and the types of data that are affected by the solution.

3 Analysis and Design

This section introduces the existing OI application, regarding its involved subjects, use cases, possible operations and types of the data. This solution is not yet GDPR compliant, so scenarios are identified for the solution along with performance, security, usability, complexity and control access requirements.

3.1 Solution introduction

The OI solution is used to give real time analytics regarding business events of targeted clients. It focuses on middleware environments, namely:

- Enterprise Application Integration (EAI).
- Enterprise Service Bus (ESB);
- Service Oriented Architecture (SOA);

This solution tracks the path from the client system to the backend systems that are invoked by SOA services, giving insight into the operation of the entire system. This insight includes metrics like reliability (percentage of errors), performance (response time), volume/load (number of requests per second) and concurrency (number of parallel requests in any instant). Figure 3.1 shows the context diagram of the OI solution.

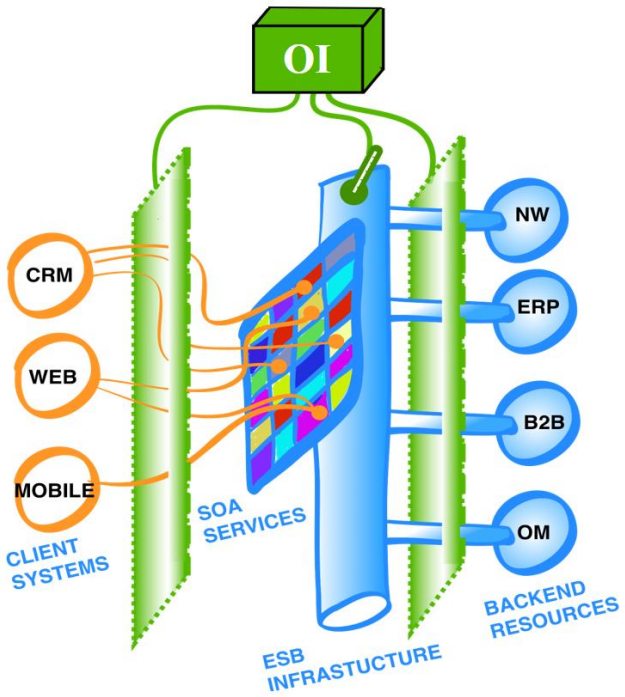

Figure 3.1 - OI solution context diagram

The OI solution is implemented in Splunk. All assumptions are made with this platform in mind, even though they're no different on other platforms. Exceptions are mentioned as needed.

3.1.1 Subjects

The OI solution is currently used by 2 types of subjects: client-side users and internal users.

a. Client-side subjects

Client-side subjects reside on clients and can only have access to certain applications of the overall product, based on the agreement with the client.

The following subjects are the most common ones on the scope of this OI solution (EAI, ESB and SOA):

Subject	Responsibilities
Application Maintenance (AM) + Service Manager (SM)	- Alarms that are related with the results of SOA service execution; - Identification of system degradation; - Root-Cause analysis.
CM	- Pre-Release vs Post-Release configuration comparison; - Alarms for frequent configuration problems.
Testers	- Test documentation and verification; - Problem resolution agility.
IT Architects	- Live documentation of the physical and logical infrastructure; - SOA Governance catalogue enrichment; - Real utilization of SOA services; - Analytic capacity to support architecture decisions.
Security specialists	- Search all accesses on assets (eg.: SOA services) by using MSISDNs or Account numbers as searching keys); - Business security.
Analysts	- SOA service analysis that is based on utilization, by historical statistics and by concrete scenario investigation (inputs, outputs, errors,…); - Transaction history analysis to support volume estimates for new processes.
Managers	- Business KPIs that are based on technical or functional data; - Real time visibility of business processes; - Analytic reporting of operations (eg.: web and call centers).
Developers	- Development of the OI solution; - Contact with all the above information, depending on the requirements that are given by administrators.
Administrators	- Responsible for all the actions of the OI solution; - Can give user access to different apps and info of the OI solution.

Table 3.1 - Client-side subjects

b. Internal users

Internal users participate on the design and implementation of the product.

Subject	Responsibilities

Developers	- Development of the OI solution; - Contact with all the above information, depending on the requirements that are given by administrators.
System Administrators	- Configuration of the OI solution; - Contact with all the above information, depending on the requirements that are given by administrators.
Administrators	- Responsible for all the actions of the OI solution; - Can give user access to different apps and info of the OI solution.

Table 3.2 - Internal user subjects

3.1.2 Use cases

Similarly to the types of subjects, the use cases can also be divided into 2 types:

- **Metrics** – used by client-side subjects;
- **Internal** – used by internal subjects.

a. Metrics

As was mentioned in section 3.1, the OI solution focuses on Reliability, Performance, Volume/Load and Concurrency metrics. These types of use cases are more commonly performed by client-side users.

Having this in mind, the following use cases are common:

Metric type	Use case	Security impact (1-5)
Reliability	Find which backend failed.	N/A
	Check which input originated the failure.	4 – Inputs may contain costumer data (which is personal more usual than not).
	Check if authentication failed in some system.	4 - May expose personal info if we're dealing with a Billing backend system for example.
Performance	Compare response times of distinct clients.	N/A
	Check if some system is slower than usual.	N/A
	Evaluate performance evolution.	N/A
Volume/Load	Find delays in batch processes.	
	Find duplicate requests.	3 – Requests may contain costumer data, although duplicate requests are not as common.
Concurrency	Count the number of parallel requests.	5 – It's almost certain that costumer data will be found on a system that works entirely in parallel due to its asynchronous nature.

Table 3.3 - Metric related use cases

b. Internal

Internal use cases are, as the name suggests, related with internal users. Internal users often need to dig deeper into the raw data that is collected by the OI application to understand how to proceed in the implementation of new features or bug fixes.

The most common are the following:

Use case	Security impact (1-5)
Get service pipelines/payloads.	5 – Service pipelines/payloads are basically the data that is in memory during the execution of some service. Service pipelines/payloads are expected to have sensitive personal information.
Get service inputs/outputs.	5 – Inputs may contain costumer data (which is personal more usual than not). This use case has more security impact than the reliability metric one due to the fact that it can be done on literally every service. On the contrary, the reliability metric use case regarding inputs only searches for the input that failed, making it less probable to find sensitive data,

Table 3.4 - Internal user related use cases

3.1.3 Operations

The OI application is expected to perform read operations on the original (unstructured) data. Data can be read directly via searches that show the raw data that was collected or via Operational Intelligence analytics that build reports and dashboards. The latter can be built by applying the operations that are show on Table 3.5.

Operation type	Description	Examples
Streaming	Operates on each event as it is returned by a search.	• eval – calculates an expression and puts the resulting value into a results field; • fields – specifies which fields should be shown/not shown; • rename – renames a field.
Transforming	Orders the search results into a data table by transforming the specified fields into numerical values.	• chart – returns the results in a table format and makes it possible to display them as charts; • stats – calculates aggregate statistics like averages, sums, etc;

		• top – returns the most common values for the specified fields; • rare - returns the rarest values for the specified fields;
Generating	Fetches information from the indexes, without any transformations.	• search; • dbinspect - returns information about Splunk indexes; • inputcsv - loads search results from a specified .csv file, which is typically a lookup).
Dataset processing	Run having in mind the entire dataset.	• sort - sorts the events based on a specified field and order; • eventstats – adds summary statistics to all results; • dedup – eliminates duplicate events.

Table 3.5 - OI solution's read operation types

No execution operations are done though. The objective of this product is to monitor system's operations and return analytic results. It's not the product's responsibility to provide tools to react upon those analytics.

3.1.4 Data

The existing OI application deals with data from the following sources:

Configuration		**Examples**
ESB configurations	Publish-Subscribe Message Delivery configurations	Publishable documents Available queues Maximum number of publishable documents
	Server configurations	Maximum DB connections Users Access control lists (ACL) Agent logs Agent configurations
Runtime configurations	Service execution metadata	Execution time Error code
	Inputs and outputs of services	Calling system Request/ESB info Personal data like: • Person name

		• Person age • Person address • …
	Infrastructure data	Available disk size Percentage of used CPU Percentage of used memory
	Server data	Connection pool Memory usage

Table 3.6 - Data types that are collected by the OI application

a. Personal Data Estimate

Of the above data types, the inputs and outputs of services are the ones where personal data might be collected. As of today, the application, namely its agent, is configured to collect all data except for personal data, due to the already mentioned penalties that are enforced by GDPR. Due to this, it's not known for certain what would be the total amount of personal data that would be collected.

For the purposes of this thesis, the amount of personal data is considered as being 1% of the total amount of data that the agent collects without the aforementioned restriction.

3.2 Personal Data

Each type of personal data can have different interpretations on the level of sensitivity it has. One might say that leaving the age of a person in clear is not as critical as doing the same with their gender or name, while at the same time the individual itself might not care about protecting his or her data, be it whatever it is.

As of today, the OI application is still not manipulating any form of personal data. Nevertheless, it is one of the objectives to start that manipulation in a near future. Thus, it is of the biggest importance that types of personal data are identified along with their proper treatment in the scope of this application and of the GDPR.

This section has the objective of identifying types of data that are expected to be collected in the future and specify an appropriate protection for them according to GDPR.

3.2.1 Types of data

For the purposes of this thesis, the following types of personal data were considered:

- Name
- Age
- Address
- Gender
- Identification Numbers (eg.: social number, billing number,…)
- Email
- Phone Number
- Birthdate

a. Application analysis

The following questions need to be answered for each type of data that was listed above:

- Is it needed? Why?
- Where can it be used on the scope of this application?
- Which security scenario is more appropriate? (Should the date be destroyed, protected, access controlled…?)

Type of data	**Is it needed?**	**Why? Where can it be used?**	**Which security scenario?**
Name	No.	The identification number is a better identifier.	Encryption in case the name is needed for some scenario in the future.
Age	Yes.	It is useful to infer the most popular age among bought products or the age where most people don't have products to attract them as clients as well.	Protection using encryption.
Address	Yes.	It is useful to infer regions where certain products are bought the most.	Not the whole address is needed, so part of it might be anonymized, while the remainder might simply be protected with encryption.
Gender			
Identification Number	Yes.	It is useful as an unique identifier of the client.	Protection using encryption.
Email	No.	Email is merely used as a contact point with the client.	Protection with encryption.
Phone Number	Yes.	Might be useful to infer the country where the client lives, if the address doesn't have that information.	Not the whole phone number is needed, so part of it might be anonymized, while the remainder might simply be protected with encryption.
Birthdate	Yes.	Might be useful to infer the age of the client, if that same field doesn't exist or was protected.	Not the whole birthdate is needed if only an approximate age is calculated, so part of it might be anonymized, while the remainder might simply be protected with encryption.

Table 3.7 - Data types analysis according to the OI application

b. GDPR analysis

Having GDPR in mind, the following questions need to be answered for each type of data that was listed above:

- Is protection mandatory?
- Is it processable according to GDPR?
- What is the appropriate protection?

Also, the answers to these questions need to have in mind the following statements about personal data:

Type of data	Is protection mandatory?	Is it processable according to GDPR?	What is the appropriate protection?
Name	Yes.	Yes.	Encryption.
Age	Yes.	Yes.	Encryption.
Address	Yes.	Yes.	Encryption.
Gender	Yes.	Yes.	Encryption.
Identification Number	Yes.	Yes.	Encryption.
Email	Yes.	Yes.	Encryption.
Phone Number	Yes.	Yes.	Encryption.
Birthdate	Yes.	Yes.	Encryption.

Table 3.8 - Data types analysis according to GDPR

All of the above types of data can be processed according to GDPR, since none of them are included on the "special categories of data" that were shown on GDPR's article 9, "Processing of special categories of personal data" (General Data Protection Regulation - Final text neatly arranged, n.d.), from section 2.1.2. Therefore, all of them can be simply pseudonymized according to the regulation.

3.2.2 Data quantity

Currently, the OI application receives approximately 24GB per day which translates to more or less 50 million events, meaning an average of 515,38 bytes per event.

3.3 Scenarios

Having in mind the analysis of the different types of personal data that were identified for this thesis, the following data security scenarios must be considered:

1. Data is so sensitive that it must not be processed or stored at all.
    a. At the moment there is no such data in the application at hand, but it might be in the future. So, for the purposes of this thesis, this scenario was also analysed.
2. Data is sensitive, but part of it might be used to infer some conclusions.
    a. Eg.: phone number prefix to infer the country.
3. Data is somewhat sensitive but it might be processed for well identified purposes
    a. Eg.: age.
4. Customer doesn't want any data to be processed, in which case everything should be discarded.
5. Customer is ok with the processing of any kind of personal data, in which case even data from the scenario 1 may be stored.

3.4 Requirements

The performance, security, usability and complexity requirements for each one of the above scenarios are here introduced. Access control requirements are also introduced here, but they're on applied to situations where users need to perform actions or execute commands to obtain the original form of the data or situations where data can be accessed on different formats on different places.

3.4.1 Performance

Requirement #	Description
P1	Data should be processed at a rate of 10MB/s or more on a regular testing environment (eg.: common desktop). This should be translated to roughly 300 events/s or 25 million events/day.
P2	In case the personal data is much slower, it should not affect the processing of the remaining data.
P3	In order to not separate the processing of personal data from the processing of regular data, the former should take a maximum of 5% more than the latter.

Table 3.9 - Performance requirements

3.4.2 Security

Scenario	Data protection	Can it be viewed and/or processed?	Where to protect the data?
1. Data is so sensitive that it must not be processed or stored at all.	Anonymization. This method corrupts the data entirely on the application.	Neither.	Indexer (Splunk side) - Anonymization Probe - Deny data forwarding for this specific type of data.
2. Data is sensitive but part of it might be used to infer some conclusions.	Format preserving Pseudonymization or a combination of encryption with anonymization, leaving part of data available for interpretation.	Partially to both.	Indexer (Splunk side) Search time (Splunk side. Requires a more strict access control policy to the server machine).
3. Data is somewhat sensitive but it might be processed for well identified purposes.	Encryption. Data should be protected, but it might be viewed in clear if needed.	Yes to both.	Indexer (Splunk side) Search time (Splunk side. Requires a more strict access control policy to the server machine).
4. Customer doesn't want any data to be processed, in which case everything should be discarded.	Anonymization. This method corrupts the data entirely on the application.	Neither.	Indexer (Splunk side) - Anonymization Probe - Deny any data forwarding.

5. Customer is ok with the processing of any kind of personal data, in which case even data from the scenario 1 may be stored.	Encryption. Data should be protected, but it might be viewed in clear if needed.	Yes to both.	Indexer (Splunk side) Search time (Splunk side. Requires a more strict access control policy to the server machine).

Table 3.10 - Security scenarios to consider in the requirements

Based on the above information, the following requirements can be gathered:

Requirement #	Description
S1	Data that falls under a "special category of data" should preferably not be forward to the OI application for processing.
S2	Data that doesn't fall under a special category of data and is only partially processed should be partially anonymized, unless a future scenario where this portion of data is needed is well identified. The non-anonymized portion of the data should be pseudonymized.
S3	Data that is processed in its entirety and doesn't fall under a special category of data, should be pseudonymized.
S4	If the customer that owns the data wishes that his personal data is not processed, all data should be immediately anonymized on the OI application. No further data should be forwarded by the probe regarding this customer.
S5	If the customer that owns the data wishes that his personal data is processed, all personal data should be pseudonymized.
S6	Both Encryptions and Anonymizations should be performed as soon as possible on the data pipeline, making full use of what the current state of art allows.

Table 3.11 - Security requirements

3.4.3 Usability

The following requirements can be gathered regarding the solution's usability:

Requirement #	Description
U1	The solution should be scalable
U2	Data should be usable after being protected.

Table 3.12 - Usability requirements

3.4.4 Complexity

The following requirements can be gathered regarding the solution's complexity:

Requirement #	Description
C1	The solution should be easily installed on clients.

Table 3.13 - Complexity requirements

3.4.5 Control Access

As was introduced on section 2.5.4 regarding access control on Splunk, Splunk comes with the administrator, power-user and user roles. Having in mind that only operations that are specifically executed by users at search time should be controlled, the following requirements can be specified:

Requirement #	Description
CA1	Only administrators should be able to execute commands that decrypt data.
CA2	All users should be able to encrypt data at search time, to protect it in the scope of an application if needed.

Table 3.14 - Control access requirements

3.5 Summary

The current OI solution focuses on the study of middleware environments like Enterprise Application Integration (EAI), Enterprise Service Bus (ESB) and Service Oriented Architecture (SOA), giving insights into performance, reliability, volume and concurrency metrics. This OI solution is implemented with Splunk, having its usual architecture, and has a probe that collects all the needed data on the targeted environments. Aside from the architecture, this section identified the involved subjects, common use cases and the operations, having in mind the criticality of all of them in security terms.

Different types of personal data that might show up on the OI application were identified here, along with multiple scenarios of data sensibility and customer will to have them being processed. Having this information in mind, performance, security, usability and complexity requirements were established for the prototypes that are proposed on section 4.

4 Protecting OI personal data

This section establishes a business scenario that is based on the personal data types that are expected to be received on the studied OI application. Since no personal data is currently being collected on the real OI application, a probe simulator was developed in order to send customer data to Splunk's Forwarder. This probe is introduced on section 4.1.2.

Different data protection mechanisms and architectures were tested having in mind the aforementioned business scenario, along with their pros and cons in terms of usability, performance, security and complexity.

4.1 Business scenario

The Business Scenario is heavily influenced by a real world OI application, also based in Splunk. This application collects both incoming and outgoing data on an Enterprise Service Bus (ESB), giving a good insight in the form of real time analytics into the data that is received from client systems, sent to the backend resources and into the data that the ESB itself generates.

For the purposes of this thesis, a simulation of the data was done instead of using the real one, since there is currently no personal data on the OI application. This also prevents any corruption of the real data that is already being used by instances of this product.

4.1.1 Simulated entities

The scenario for the presented prototypes is a telecommunications company. This company sells subscriptions that can have different price plans, products like phones, sim cards, and so on.

The existing entities can be found on Figure 4.1.

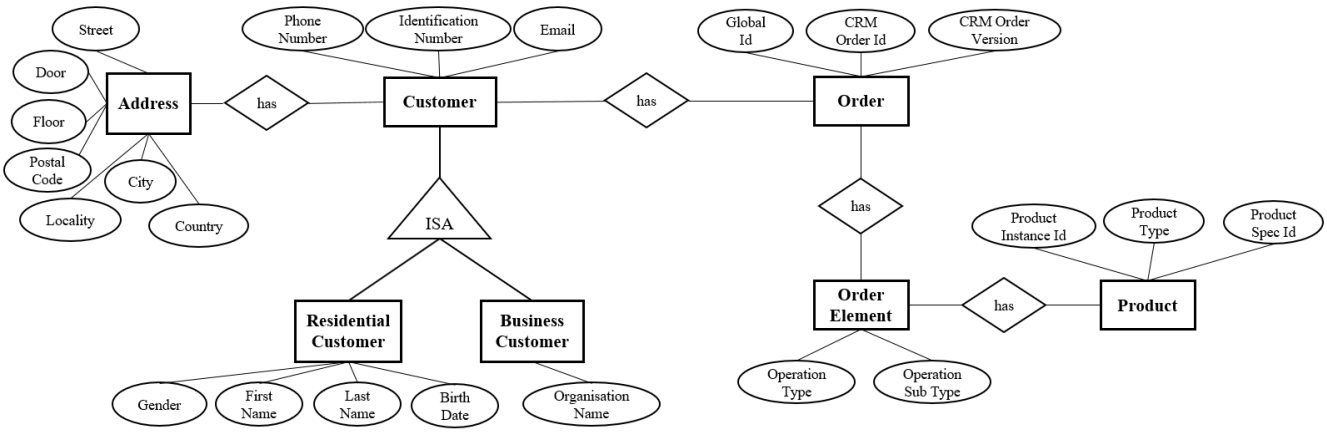

Figure 4.1 - Entity diagram of the prototypes' simulated data

The actual orders are as simple as it gets as the main purpose of this model is to have clients with realistic personal data that, in a common scenario, would be found on Splunk logs.

4.1.2 Splunk Probe simulator

This probe is implemented in Java and is merely a library to generate random customer data. All data is exported to a log file that is then read by the Splunk forwarder.

The generated customer data has the following format:

"*<TIMESTAMP>,<GLOBAL_ID>,<CRM_ORDER_ID>,<CRM_ORDER_VERSION>,<ADDRESS>,<IDENTIFICATION_NUMBER>,<EMAIL>,<PHONE_NUMBER>,<CUSTOMER_NAME>,<CUSTOMER_FIRSTNAME>,<CUSTOMER_LASTNAME>,<CUSTOMER_TYPE>,<BIRTHDATE>,<GENDER>,<OPERATION_TYPE>,<OPERATION_SUB_TYPE>,<PRODUCT_TYPE>,<PRODUCT_INSTANCE_ID>,<PRODUCT_SPEC_ID>*"

The above format represents a single element of a customer order, which means that, most of the times, there is more than one line for each global id, customer id and client info (name, email, etc.), which is to be expected in most big data scenarios.

4.2 Prototypes

The following four prototypes were tested:

1. Prototype 1: Using built-in anonymization methods to protect the data.
2. Prototype 2: Using 2 separate indexes on Splunk, one with protected data and another with data in clear text.
3. Prototype 3: Using a Python library for the Paillier homomorphic encryption scheme.
4. Prototype 4: Using Microsoft's SEAL library for fully homomorphic encryption.

4.2.1 Prototype 1 - Anonymization

On this prototype, the data was here protected once it reaches the Splunk indexer through the use of anonymization techniques.

a. Prototype architecture

This prototype follows a standard Splunk architecture, where we have a Splunk forwarder, a Splunk Indexer and a Splunk Search head, like shown on Figure 4.2:

- Splunk forwarder – where the data originates from in clear text.
- Splunk indexer – where the data is stored and where the data is protected.
- Splunk search head – where the data is read by end users.

The data is generated randomly by the Probe simulator that was mentioned on section 4.1.2.

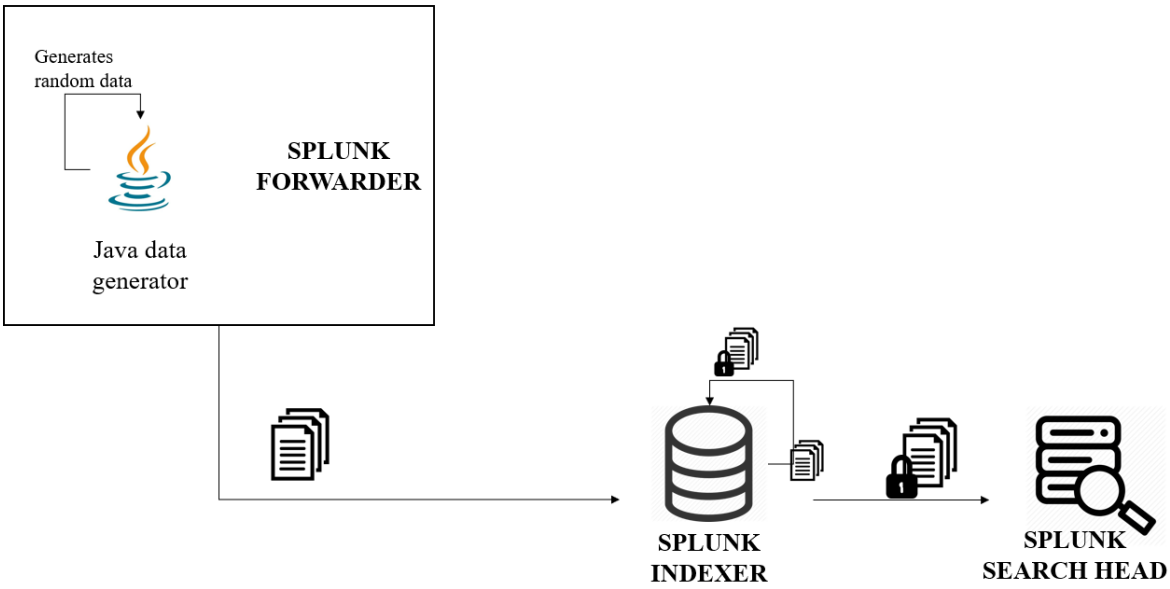

Figure 4.2 - Anonymization prototype architecture

b. Data protection

The data is protected at “index-time” by using regex commands (see Annex A) to identify personal data and replacing the results with non-sense values. This method is called “Regex Replace” and was already introduced on section 2.5.3.

The Regex Replace method implements data anonymization, thus making the data not reversable. Figure 4.3 shows an example of the logs that Splunk shows after the fields are protected:

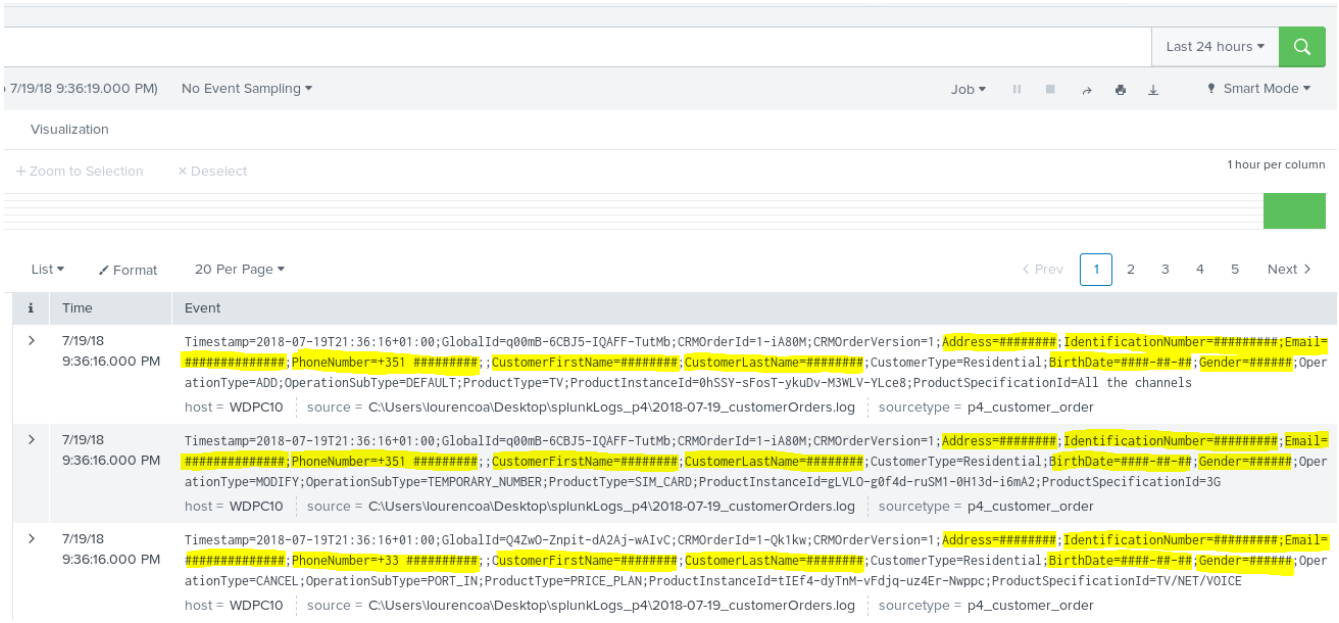

Figure 4.3 - Anonymization data protection

c. Access control

This solution doesn't need any kind of access control to personal data, since the data is already protected in such a way that making it able to seen or not to certain users doesn't make any difference.

d. Prototype analysis

This prototype was here analysed regarding performance, security, usability and complexity.

Being a regex replace based prototype, it is expected that it shares similar characteristics to ones shown by the "Regex replace" option of section 2.5.3. It has indeed high security, acceptable performance and low complexity as good characteristics, while having low usability as a bad characteristic. These are analysed in more detail in the following sections.

i. Performance

In terms of performance, this solution is as best as it can get when it comes to field protection on the Splunk data layer. There is no computation involved in the field protection other than finding the field's value and replacing it some value.

The part of the latency that results from data forwarding can be ignored on this prototype, since no computation is performed on the data before reaching the Splunk indexer. Only the difference between the first and last event index time needs to be considered here.

3 load tests were performed with this solution:

Test #	Number of customer order (distinct global ids)	Number of events	Total index time (secs)	Avg. time per event (secs)
1	1000	11058	76	0,00687
2	5000	49969	106	0,00212
3	10000	101329	218	0,00215

Table 4.1 - Anonymization's performance metrics

ii. Security

In terms of security, this solution is also the best that can be achieved. There is no way of recovering the personal data since, in practice, the data is being destroyed.

iii. Usability

In terms of usability, this solution as bad as it gets. It is impossible to use the personal data in any way since, again, the data was essentially destroyed.

The data is only slightly usable, without jeopardizing the security, if some part of it is left in the original form. Some examples are:

- One can leave just the phone number prefix in clear in order to conclude to which country the phone number belongs.
- One can anonymize the phone number of a client but leave the number of digit evident (eg.: if the phone number is replaced by 9 digits with no meaning, one could conclude that this number is from Portugal if that's the only country with 9 digits on the company).

Both methods were used on the protype at the same time, as can be seen on Figure 4.4.

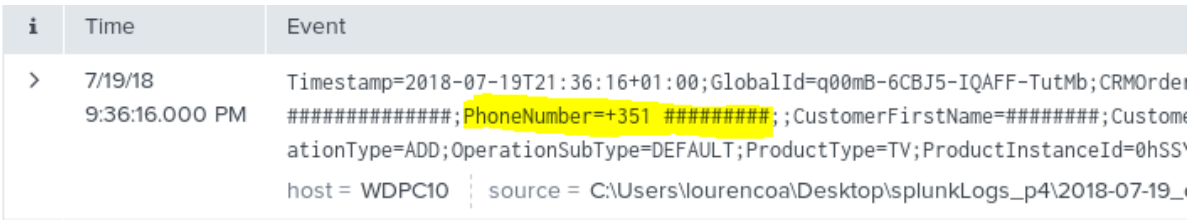

Figure 4.4 - Anonymization prototype's usability

iv. Complexity

This solution is based on building appropriate regex commands and typing a meaningless string to replace the resulting value, which can be considered as simple. A way of deploying along with the solution to a client would be to have a simple configuration file containing the regular expressions, which is already provided by Splunk.

e. GDPR analysis

There are two situations to be considered when analysing whether or not this solution is compliant with GDPR, namely the following two articles:

- **Article 17 – Right to be forgotten** – It's here mentioned that any data subject can have all data concerning him erased without any delay (General Data Protection Regulation - Final text neatly arranged, n.d.).
- **Article 32 – Security of processing** – It's here mentioned that all stored personal data should be protected according to its level of risk and its integrity, among other security principles, should be enforced (General Data Protection Regulation - Final text neatly arranged, n.d.).

Article 17 shows a good scenario where this solution can be applied. Just as was introduced on the security requirements, namely requirement S4, the data subject can ask for all his/her data to be removed or destroyed, and this can be easily achieved with anonymization.

Article 32 shows the opposite. If the anonymization is used solely as a mean to protect a specific portion of data, that same portion of data will have its integrity compromised, thus compromising the following rights of the data subject as well:

- **Right of access (article 15),** which states that the data subject should be able to access his or her personal data (General Data Protection Regulation - Final text neatly arranged, n.d.);
- **Right to rectification (article 16),** which states that the data subject should be able to rectify his or her personal data (General Data Protection Regulation - Final text neatly arranged, n.d.);
- **Right to data portability (article 20),** which states that the data subject should be able to receive the personal data concerning him or her in a structured format (General Data Protection Regulation - Final text neatly arranged, n.d.).

If the data is destroyed with anonymization, the user will not be able to access it, rectify it (since the data is not understandable) or received it in a structured format. On might add that destroyed data might mean that the user has no data concerning him or her in the first place, but with operational intelligence solutions this data is originated from a lot of systems meaning that it is through these solutions that the data is obtained most of the time.

f. Prototype conclusion

This prototype has very clear performance, complexity and security advantages since, due to the low load that is induced on the solution to protect the data, the protection of data is very efficient, very fast and easy to implement.

Nevertheless, all the good things about this prototype are masked by its complete lack of usability. Replacing data by meaningless values is pretty much the same as destroying the data which, in term, is the same as not having it at all. The whole point of protecting personal data instead of deleting it right away is to allow an analysis to be performed upon that data latter.

This prototype is not advisable in a context where there is no way of cross-referencing the anonymized data via non-anonymized fields.

Finally, this solution should never be used as a means to protect personal data that the user didn't ask to be forgotten. If the data exists, which happens on a lot of the systems that are monitored with operational intelligence, it should be easily obtained, normally via the OI solution itself. The only scenario where this solution should be used is when the data subject actually invokes his right to be forgotten.

Section 4.2.2 introduces an alternative mechanism called AES, which stands for Advanced Encryption Standard. AES offers a better way of protecting data for scenarios where the user didn't ask to be forgotten, like said above.

4.2.2 Prototype 2 – AES

On this prototype, the data was protected with the AES encryption scheme on one of two places:

- **On an external processor, before reaching the indexer** – This approach requires an additional server to be installed/deployed but ensures that the data is stored already on a protected state.
- **At search time** – This approach doesn't need an additional server to be installed/deployed but leaves the data on an unprotected state.

Both approaches are not perfect as was explained above. The reason why an approach that protects data at index time wasn't adopted is that there is no such approach on Splunk's current state of art.

After the encryption is performed, the data is only visible on clear text by users with enough permissions that allow them to execute the appropriate decryption command on Splunk. Both decryption and encryption commands were implemented in Python as part of this prototype. In this case, users with higher permissions (eg.: administrators) are able to decrypt the data while users with lower permissions (eg.: developers or users) are only able to encrypt the data if needed.

a. Prototype architecture

As was already said, this prototype may follow one of the architectures that are shown on Figures 4.5 and 4.6:

i. External processor

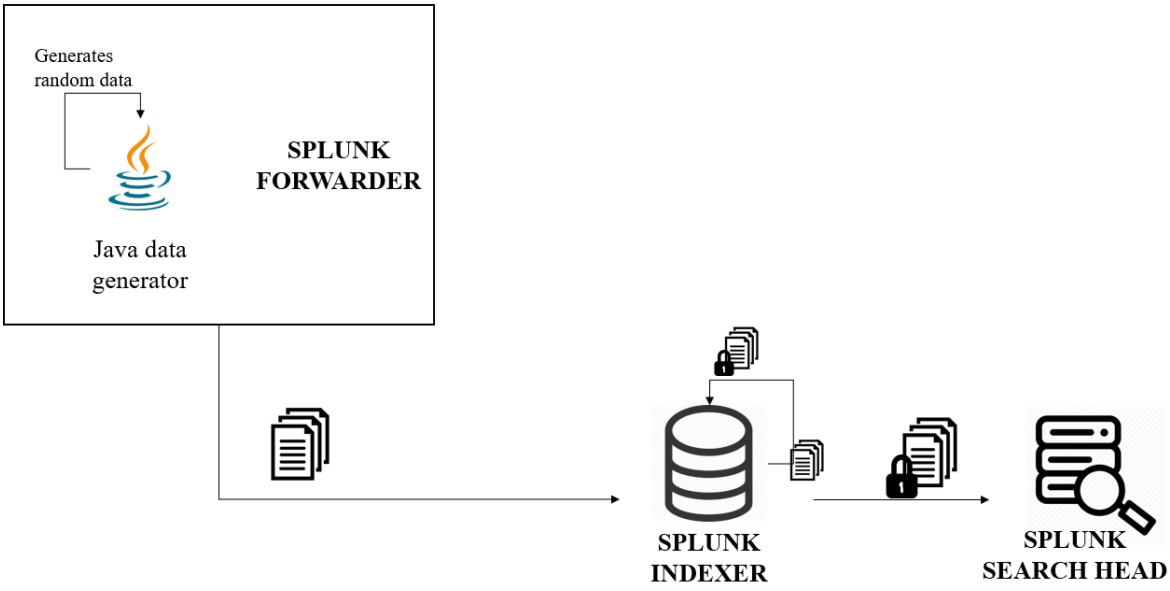

Figure 4.5 - AES prototype's external processor architecture

ii. Search time

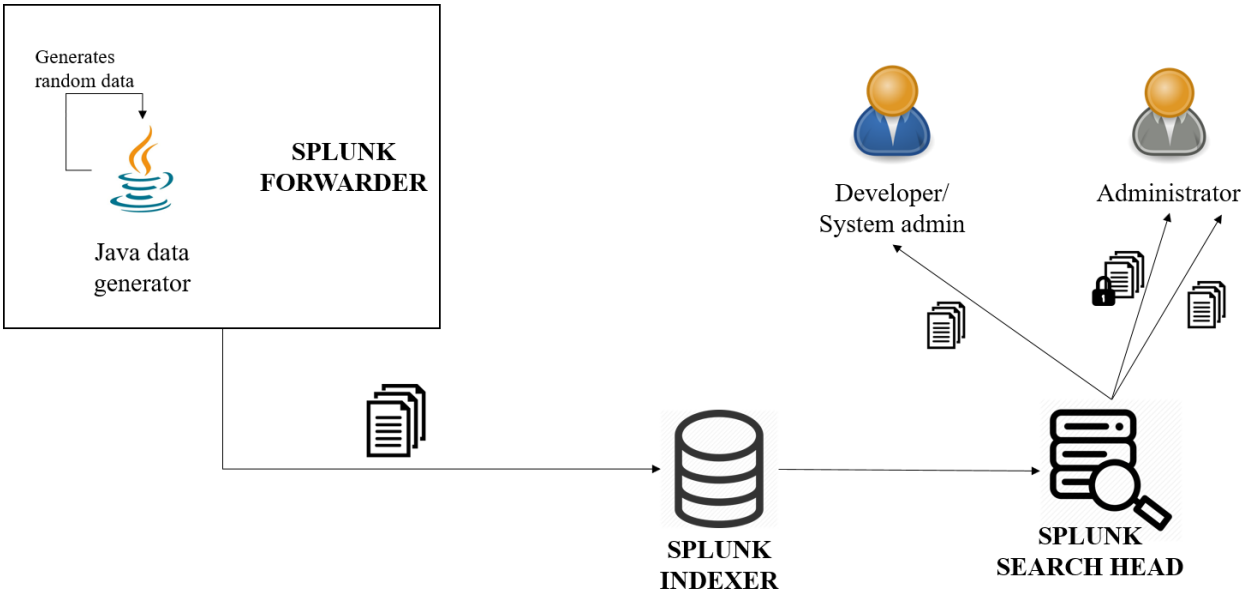

Figure 4.6 - AES prototype's search time architecture

b. Data protection

The data can be protected on an external processor or at search time. For the purposes of this prototype, the AES algorithm was applied to all available personal data, in order to get the full impact of this solution.

Figure 4.7 shows an event with data that was already protected with AES:

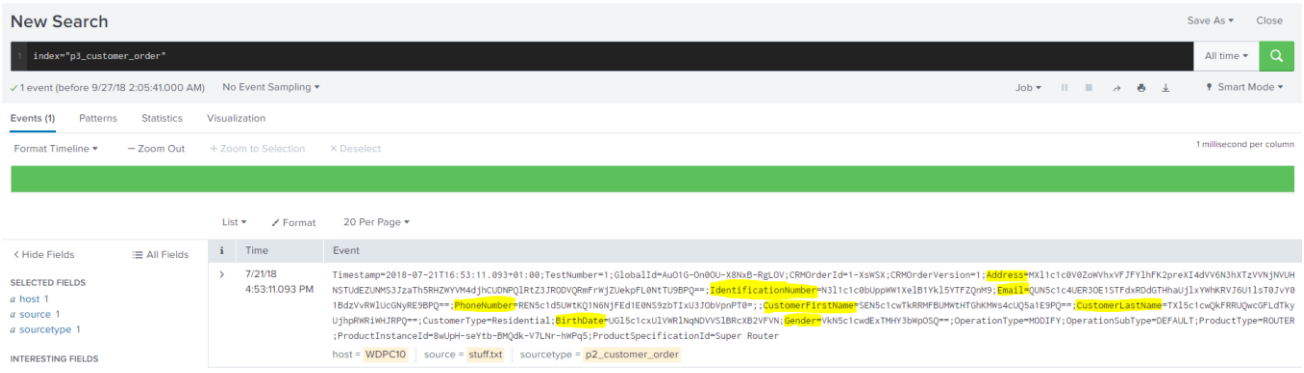

Figure 4.7 - Data protection of the Paillier prototype

These same fields can later on be decrypted by users with enough permissions with python commands on the Splunk side.

For this purpose, a command called “decryptaes” was created. This command receives field names as arguments and decrypts them using a secret key that is shared between the indexer and the external processor.

As said before, the data can also be encrypted at search time. For this purpose, a “encryptaes” command was also created. The result of this command is exactly the same that is shown on Figure 4.7.

As an example, let’s take the “BirthDate” field:

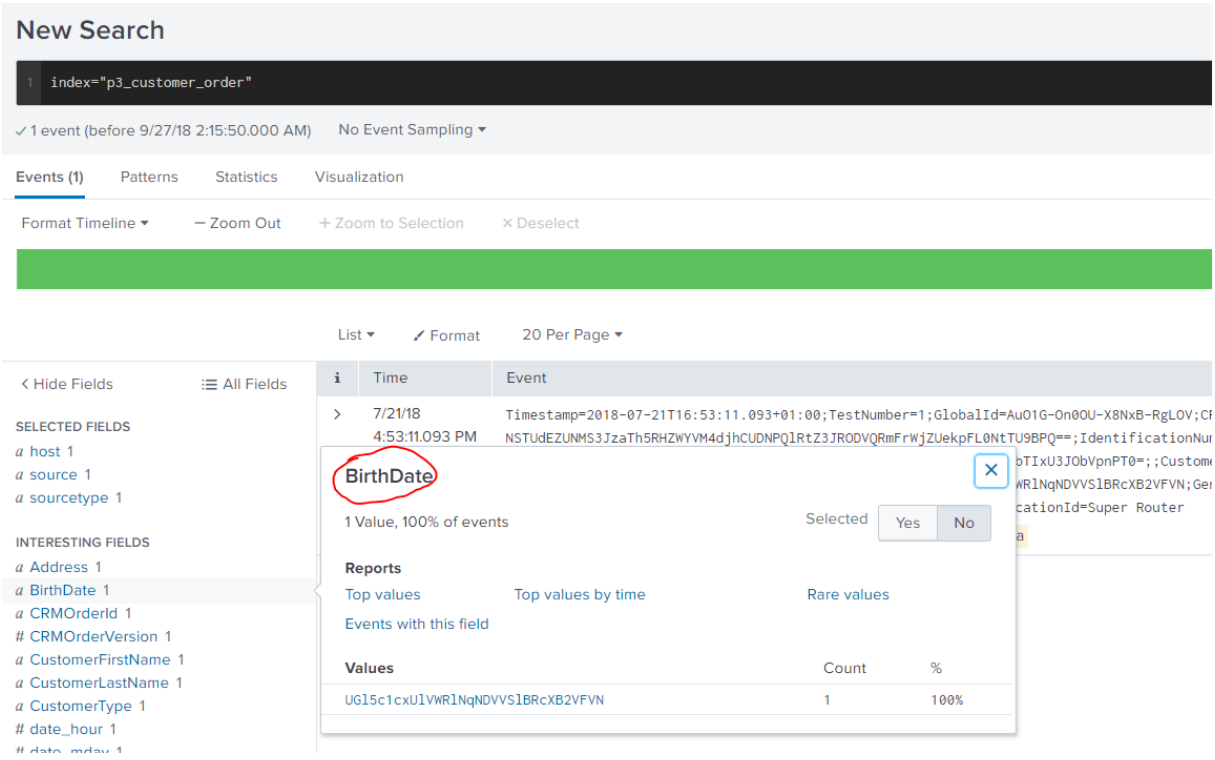

Figure 4.8 – Field encryption of AES

As can be seen on Figure 4.8, the field contains an already encrypted value. Figure 4.9 shows the application of the “decryptaes” command into the encrypted value using the Administrator account:

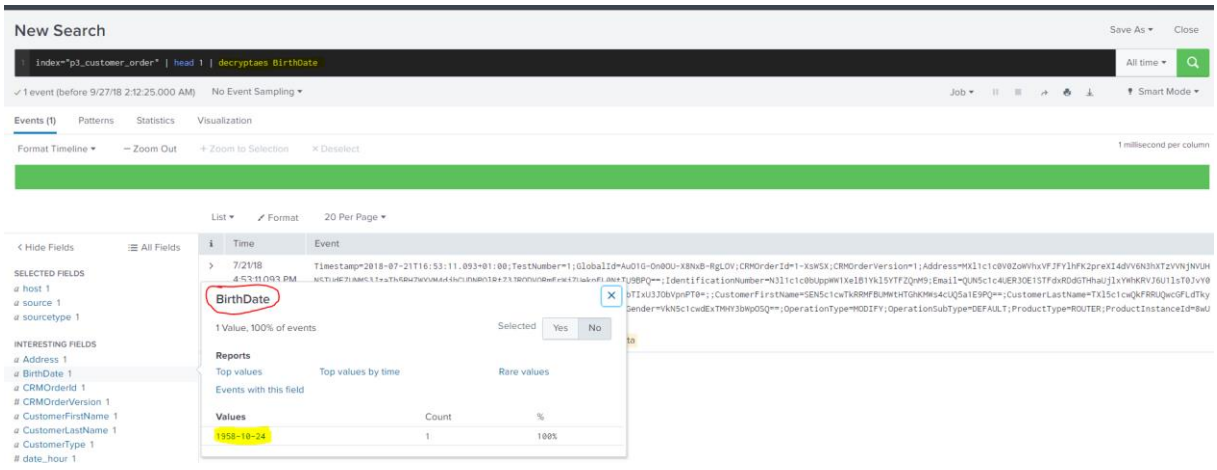

Figure 4.9 – Field decryption of AES

c. Access control

This prototype follows a simple Role-Based Access Control (RoBAC) where users with different levels of privileges are able to see different types of data, reports, dashboards and so on. In this matter, users can be split into 2 categories on this prototype:

- Users that can see personal data via searches that make use of a decryption command.
- Users that can only see the protected version of the customer data via searches that make use of an encryption command.

The way these generic users types are fit into the subject types that were introduced on section 3.1.1 is explained below.

i. Client-side subjects

Only the following client-side subjects should be able to execute the decryption command to see personal data in clear-text:

Subject	Examples of personal data usage
Application Maintenance (AM) + Service Manager (SM)	Root-cause analysis for eventual corrupted data, like unexpected values for certain users for example.
Security specialists	This subject is responsible for ensuring the security of the application, hence, it's also his/her responsibility to understand if personal data is corrupted for example.
Administrators	Last resort user.

Table 4.2 - AES's client-side access control

As can be seen of the following section, the main difference between client-side subjects and internal users is that developers are not allowed to see personal data in clear-text. That responsibility should be passed to the internal users or, in a more critical situation, to security specialists.

ii. Internal users

All internal users but System Administrators should have access to the decryption command in order to implement/manage the OI solution properly. The following examples showcase what each user should be able to do with personal data:

Subject	Examples of personal data usage

Developers	Implementation of dashboards that analyse personal data like to conclude: - average of age - most popular gender for each product type - …
System Administrators	N/A.
Administrators	Analyse available personal data against the current situation of the company in order to specify requirements for the Developers.

Table 4.3 - AES's internal user access control

d. Prototype analysis

This prototype is here analysed regarding performance, security, usability and complexity.

From the field protection options that were introduced on section 2.5.3, the "External Processor" and "Search time" approaches were tested on this prototype. The "Modular input" and "Scheduled search" approaches could've also been tested, but the principle of executing an additional python script would be the same, thus inducing the same kind of overhead. The "Result Masking" would've not protected the actual data that is stored on the indexer, since the protection is applied only at search time on this approach.

i. Performance

Tables 4.4 and 4.5 show the following performance information:

- Both encryption start and end timestamps were sent to the Splunk indexer along with the already protected data.
- 3 tests were performed:
  - Test 1 – 1000 customer orders
  - Test 2 – 5000 customer orders
  - Test 3 – 10000 customer orders

The performance of AES using the "External Processor" approach is shown on Table 4.10.

Test #	**Number of customer order (distinct global ids)**	**Number of events**	**Bytes**	**Total encryption time (secs)**	**Avg. time per event (secs)**	**Avg. time per MB (secs)**
1	1000	9987	5 889 478	322,71	0,032	0.0548
2	5000	50120	26 943 556	1476,36	0,029	0.0548
3	10000	100927	56 886 143	3270,81	0.032	0.0575

Table 4.4 - AES performance metrics – External processor approach

The performance of AES using the "Search time" approach is shown on Table 4.11.

Test #	**Number of customer order (distinct global ids)**	**Number of events**	**Bytes**	**Total encryption time (secs)**	**Avg. time per event (secs)**	**Avg. time per MB (secs)**
1	1000	9899	6 463 408	354,16	0.035	0.0548
2	5000	50345	32 831 826	1798,45	0.036	0.0548

3	10000	100521	65 698 287	3600,36	0.036	0.0548

Table 4.5 - AES performance metrics – Search time approach

As is shown on Table 4.10, this prototype loses pretty much a decimal value in comparison with the anonymization prototype. Nevertheless, it still encrypts all personal data of a single event in approximately 0,03 seconds and encrypts 1 MB in approximately 0.05 secs.

Taking into account the personal data estimate of 1% that was established on section 3.1.4 and considering that approximately 24GBs of data (~50 million events) are received every day, the OI application is expected to receive 240MBs of personal data per day.

In conclusion, 12 seconds would be needed to encrypt all personal data of a single day, which is more than acceptable.

ii. Security

Of the two provided architectures, the External Processor is the best one in terms of security, since it protects all personal data before it reaches the Splunk solution environment. Data can then be seen in its clear text format only by users with high enough privileges on the access control rules, which in this case are the Administrators. This solution meets security requirements 3, 5 and 6 and also partially meets requirement 2, which is a mixture of this solution and the anonymization one (see section 4.2.1).

On the other hand, the "Search Time" solution protects the search tier only and leaves the data stored in the indexer on an unprotected format. This solution also meets requirements 2 (partially), 3, 4 and 5, but requires that the server should only be accessed locally by users with high privileges (administrators and system administrators). The usability problems that originate from this approach are explained on the Usability section.

iii. Usability

Both "External Processor" and "Search time" have the same problem of usability in the sense that data can no longer be processed once it is encrypted, unless an user with high enough privileges comes into the picture. The only data processing that normal users (eg.: developers) can perform are string concatenations, and even that operations requires full documentation of the type of data and approval from the administrators to ensure that there is no corruption and that the data is on the intended format.

On top of this, the "Search Time" approach also requires that no users other than Administrators and System Administrators can access the server locally, due to the data being stored on clear text, which is historically difficult to implement.

Both solutions meet requirements U1 and U2, even though only partially. According to U1, the solution should be scalable, which forces roles to be well defined onto each users that joins the application. According to U2, data should be usable, which is fully met for Administrator users only.

iv. Complexity

The "External processor" approach has the problem of requiring an additional server, which will probably not be attractive to a client.

Also, it requires additional decryption commands to be deployed with the application as well, being that the decryption must have an enforced strict access control. This command also needs a secret key to be shared between the external processor server and the indexer server, which in itself already requires constant maintenance. Maintaining this same secret key on clustered environments exponentiates the complexity of the solution even more.

The “Search time” approach avoids the problem of shared keys between servers, which makes it less complex, and avoids the problem of having an additional server. Nevertheless, it has the same problem of access control for the encryption and decryption commands.

e. GDPR analysis

The “External processor” approach complies with everything that is stated on GDPR. All personal data is protected with the use of encryption - like said on GDPR’s article 25 “Data protection by default ad by design (General Data Protection Regulation - Final text neatly arranged, n.d.), of section 2.1.4 - before it is stored on the indexer machine.

As for the “Search Time” approach the same cannot be said. The data is left on a clear text format on the indexer, meaning that anyone with enough privileges can access the server locally and access the data on its original format. This breaks the same article 25 that was mentioned above.

Both approaches maintain the integrity of the data - like said on GDPR’s article 5 (General Data Protection Regulation - Final text neatly arranged, n.d.), “Processing of personal data”, of section 2.1.2 - since a user with enough privileges can still use a decryption command to see data in its original form. The only kind of data processing that can be done is to concatenate strings at search time, which still doesn’t affect the actual data that is stored on the indexer anyway.

f. Prototype conclusion

This prototype uses a symmetric encryption scheme named AES that pseudonymizes the data. This prototype may be implemented with the help of an “External Processor” server that protects the data before it reaches the indexer or with encryption commands that are executed at search time.

None of the approaches is perfect, being that the External Processor is weak in usability and complexity but strong on security and fully compliant with GDPR and the approach at Search time is strong in complexity, less weak in usability but with discussable compliance with GDPR.

Both approaches may be used depending on the scenario at hand. A client with fewer resources and capacity but strict when it comes to security might be more keen on using a Search time approach, while a client who is able to spend more resources on a single solution might choose the External Processor approach and allocate more people into the project to properly maintain the solution.

In terms of performance, this solution is not as good as the Anonymization one but it is still more than acceptable, offering encryption of data equivalent to one day in approximately 12 seconds. Like is shown on sections 4.2.3 and 4.2.4, it surpasses by far both Paillier and SEAL, respectively.

Like was mentioned above, sections 4.2.3 and 4.2.4 put into the test Paillier and SEAL, which are homomorphic encryption mechanisms. This type of encryption allows the data to be processed with a set of operations even when this same data is already encrypted, thus offering a good solution to the usability problem of AES.

4.2.3 Prototype 3 – Paillier

The Paillier library is implemented in Python. As was shown in the Paillier dedicated part of section 2.3.3, the homomorphic property of this scheme is:

$$\mathcal{E}(x1).\mathcal{E}(x2) = (g^{x_1} r_1^{m})\,(g^{x_2} r_2^{m}) \bmod m^2 = g^{x_1+x_2}\,(r_1 r_2)^{m} \bmod m^2 = \mathcal{E}(x1 + x2)$$

In practice, this means that Paillier is well suited for add and multiply operations. Being Splunk and, more generically, operational intelligence areas that deal with alphanumerical data, this algorithm is not enough as is, since it works only for numeric values.

Nevertheless, it can be adapted to work with strings as well by converting them to Unicode and performing the operations on that value. For the purposes of this document, only the concatenation operation was implemented to test this scheme on a close to real environment.

a. Prototype architecture

This prototype follows an architecture with an external processor located between the Splunk forwarder and the Splunk Indexer. This external processor is where the Paillier algorithm is executed to protect the sensitive personal data before it reaches the Splunk indexer.

Figure 4.10 illustrates the architecture described above.

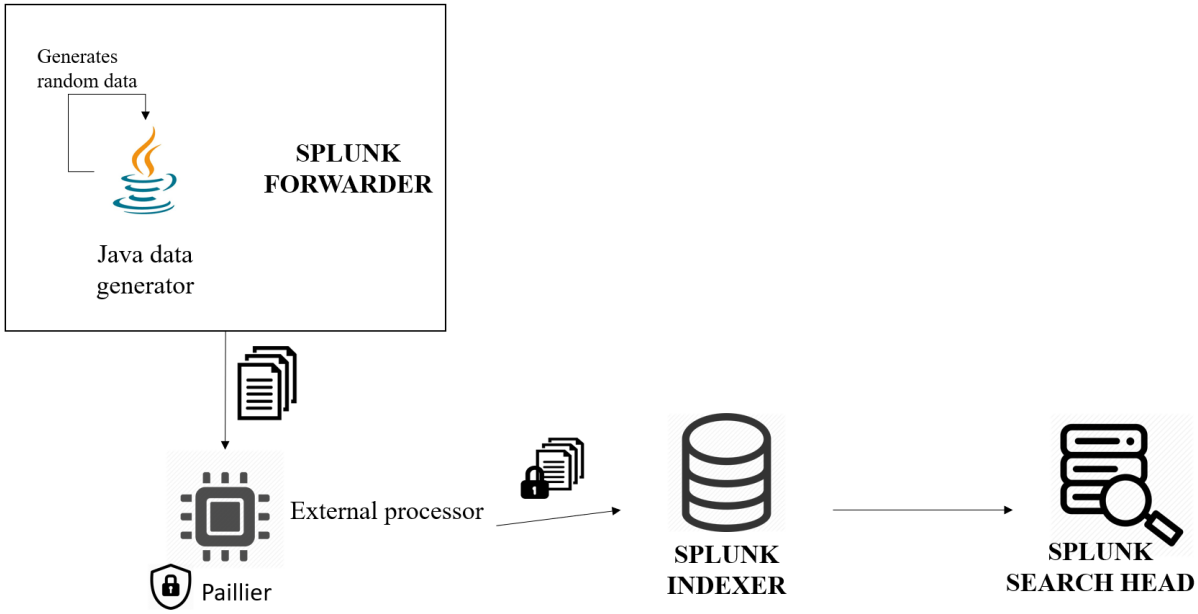

Figure 4.10 - Paillier prototype architecture

b. Concatenation adaptation

Paillier is, like most homomorphic encryption schemes, an algorithm that allows processing of numeric data only. Nevertheless, the studied OI solution has mostly alpha-numeric values.

For the purposes of this thesis, this algorithm was adapted to allow string concatenations which is one of the most common operations and also the easiest to implement. The purpose of using concatenations is to showcase the potential for evolution in homomorphic encryption.

In order to adapt the Paillier algorithm to String concatenations, the following logic was implemented for encryptions:

1. Both strings s1 and s2 are converted to Unicode.
2. Both unicodes U(s1) and U(s2) are encrypted using the encryption code that is provided by the Paillier library.

3. The encrypted unicodes E(U(s1)) and E(U(s2)) are concatenated with a secret split character between them.

For decryptions:

1. The encrypted data E(U(s1)) ⊕ E(U(s2)) is split by the secret split character that was previously defined, originating two different encrypted strings E(U(s1)) and E(U(s2)).
2. Both encrypted strings are decrypted originating the original Unicode values U(s1) and U(s2)
3. The Unicode values are concatenated back originating U(s1⊕ s2)
4. The Unicode value U(s1⊕ s2) is converted back to String originating s1⊕ s2.

The whole idea behind this algorithm is quite simple and it relies solely on the split character, which must be safe. In order to ensure its safety, it should be encrypted as well and should remain a secret to the application, in the same way as a secret key.

i. Code

In order to verify the above logic, 2 Python classes were created:

- **StringPaillier.py** – This class implements all the needed methods to convert String to Unicode, convert Unicode to String, concatenate encrypted strings and decrypt concatenated strings.
- **Test.py** – This is the main class. It asks the user for 2 strings that he wishes to concatenate, encrypts both strings, uses the secret split character to concatenate them and finally reverts the encryption. All this is done with the methods from the StringPaillier class and the results are shown on the terminal.

The actual code for the above files can be found on Appendix A.

ii. Execution

In the following scenario, the user chose strings "Andre " and "Lourenco" and, as can be seen on yellow, the result is the expected one, i.e. "Andre Lourenco". Please note that the encrypted values on Figure 4.11 are not shown on its entirety, since they were to big to fit in the screen.

```
<terminated> Test.py [C:\Python27\python.exe]
2018-06-14 11:35:07: Generating keys...
2018-06-14 11:35:08: Keys generated!
Available operations:
  add
  subtract
  concat
Choose your operation: concat
String 1: Andre
String 2: Lourenco
2018-06-14 11:35:22: Encrypting Str1...
2018-06-14 11:35:28: Str1Encrypted!
2018-06-14 11:35:28: Encrypting Str2...
2018-06-14 11:35:33: Str2Encrypted!
2018-06-14 11:35:33: Decrypting...
2018-06-14 11:35:33: Decrypted!
2018-06-14 11:35:33: Decrypting...
2018-06-14 11:35:33: Decrypted!

Here's your result:
  String 1 encrypted: 17447486169202756170320603176055240514942948074740796426485629571717671606708682781255145948022779990745071775471883401487780721503642400371
  String 2 encrypted: 91085589694112724162988632578284351249770876221824998715120093282036538013497877377310011787075595484557096379267635355438729127651
  encrypted: 17447486169202756170320603176055240514942948074740796426485629571717671606708682781255145948022779990745071775471883401487780721503642400371
  decrypted (unicode): 065110100114101032076111117114101110099111
  decrypted (String): Andre Lourenco
```

Figure 4.11 - Execution of the Paillier concatenation adaptation

It's also worth noting that the encryption took close to 5 seconds per each String. Table 4.6 shows the execution time for 7 different string concatenations.

String 1	String 2	Encryption time (string 1)	Encryption time (string 2)
“Andre “	“Lourenco”	5,260 secs	5,074 secs
“Cristiano “	“Ronaldo”	5,012 secs	4,460 secs
“AaBbCcDdEeFfGgHh”	“IiJjKkLlMmNnOoPp”	4,865 secs	5,176 secs
“qwertyQWERTYqwerty”	“QWERTYqwertyQWERTYqwerty”	5,090 secs	4,959 secs
“Hello “	“World”	4,774 secs	4,631 secs
“221B Baker Street“	“, London, England”	4,872 secs	5,410 secs
“abdefghijklmnopqrstuvwxyz”	“ABCDEFGHIJKLMNOP QRSTUVWXYZ”	4,738 secs	5,146 secs

Table 4.6 - Paillier's concatenation adaptation execution times

This is not ideal on Operational Intelligence solutions, where it is expected that dozens of gigabytes are received per day.

c. Data protection

The data is protected on an external processor, before reaching Splunk territory. For the purposes of this prototype, all personal data was protected with the Paillier algorithm, in order to get the full impact of this solution. The result of this protection is shown on Figure 4.12.

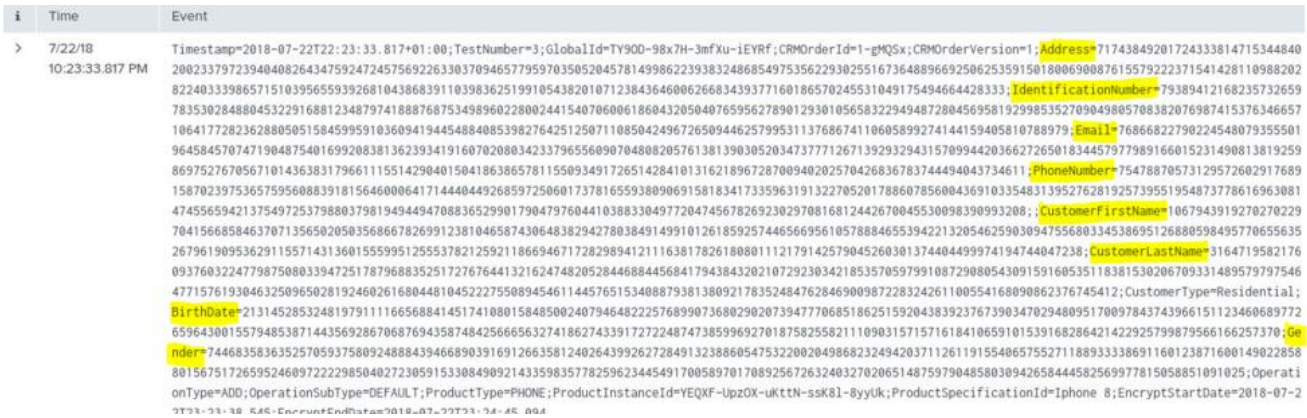

Figure 4.12 - Data protection of the Paillier prototype

This same fields can later on be decrypted by users with enough permissions with python commands on the Splunk side.

For this purpose, a command called “decryptpaillier” was created. This command receives field names as arguments and decrypts them using the public and private keys from the Paillier algorithm that are shared between the indexer and the external processor.

As an example, let’s take the “BirthDate” field:

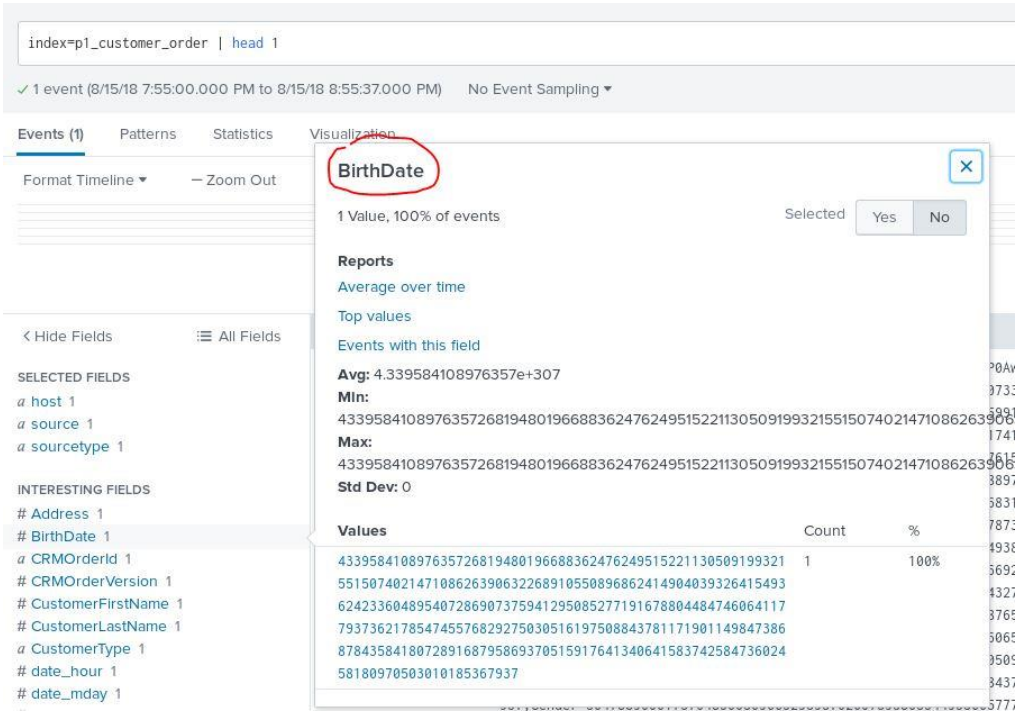

Figure 4.13 - Paillier's field encryption

As can be seen on Figure 4.13, the field contains an already encrypted value. Figure 4.14 shows the application of the “decryptpaillier” command to the encrypted value using the Administrator account:

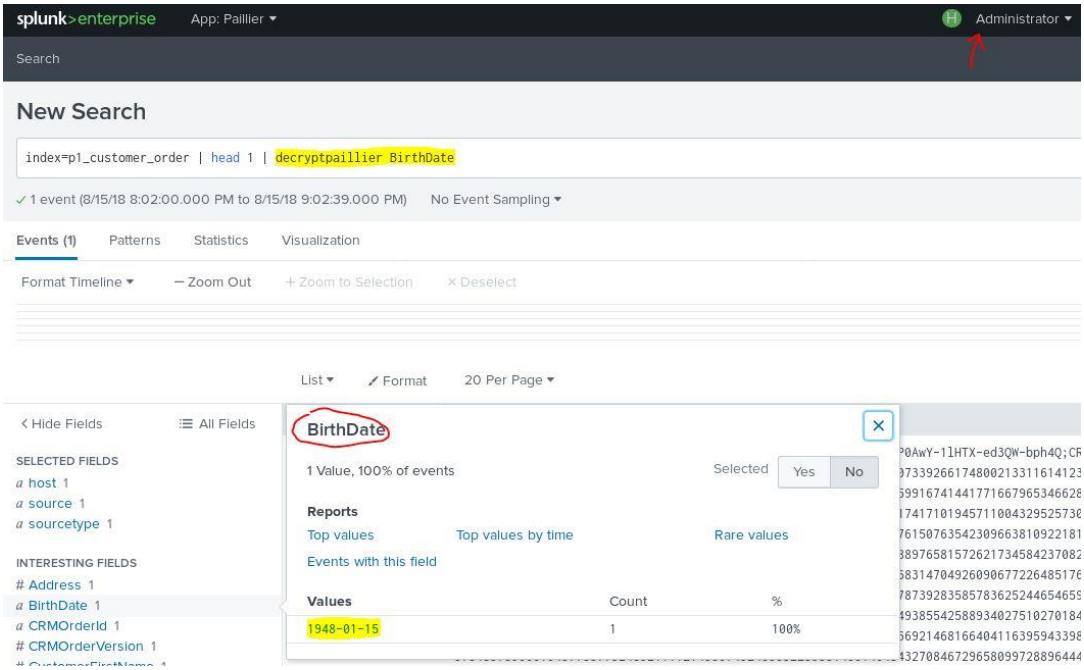

Figure 4.14 - Paillier's field decryption

d. Access control

A Role-Based Access Control policy was applied to the decryption command that was implemented on Splunk. The command was configured to be readable by the admin user role only. The admin role here is equivalent to the “Administrator” role that was introduced on the Internal users of section 3.1.1.

The decryption was already shown on the previous section. Nevertheless, if a user outside the admin role were to try to use this command, Splunk acts as is the command doesn't exist at all, like shown on Figure 4.15.

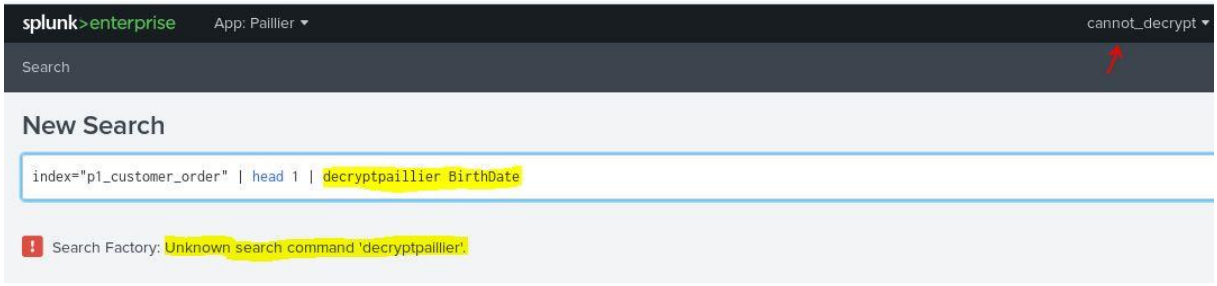

Figure 4.15 - Paillier's access control

e. Prototype analysis

This prototype is here analysed regarding performance, security, usability and complexity.

From the options that were introduced on section 2.5.3 for data obfuscation and field protection, the "External Processor approach is tested on this prototype. The "Modular input" and "Scheduled search" approaches could've also been tested, but the principle of executing an additional python script would be the same, thus inducing the same kind of overhead. The "Result Masking" would've not protected the actual data that is stored on the indexer, since the protection is applied only at search time on this approach.

i. Performance

As was already shown on the "Concatenation adaptation" section of this prototype, data encryption/decryption takes too much time for an Operational Intelligence solution. It takes approximately 5 seconds to encrypt 1 single string, with the increased handicap that this approach requires an additional machine to communicate with.

Table 4.7 proves the above statement with the following information:

- Both encryption start and end timestamps were sent to the Splunk indexer along with the already protected data.
- 3 tests were performed:
  - Test 1 – 1 customer order
  - Test 2 – 3 customer orders
  - Test 3 – 5 customer orders

Test #	Number of customer order (distinct global ids)	Number of events	Size (bytes)	Total encryption time (secs)	Avg. time per event (secs)	Avg. time per MB (secs)
1	1	19	10608	1038.308	54.647	97,88
2	3	29	16248	1486.541	51.260	91,49
3	5	60	31567	3669.511	61.158	116,25

Table 4.7 - Paillier's performance metrics

As expected, Paillier's performance is completely unacceptable. It takes approximately 55 seconds to protect one single event with 8 personal data fields and approximately 100 seconds to protect 1MB. This

makes the implementation of this solution impracticable in a scenario where approximately 24GBs of data or 50 million events are received each day.

Even with the estimates of having 1% of the whole data being personal (as established on section 3.1.4), this leaves approximately 240MBs that need to be protected, which translates to 24000 seconds or 6,6 hours of data protection.

ii. Security

This solution can be applied on scenario 2 with combined encryption and anonymization, scenario 3 with pure encryption for specific data and scenario 5 with pure encryption for all data.

This solution is pretty good security wise, in the sense that the protected data (or portion of data) cannot be interpreted by the human eye without being decrypted, just like any normal encryption scenario.

iii. Usability

Even though the data is pseudonymized, it can still be processed for certain operations. Nevertheless, the amount of work that can be done on the data without jeopardizing its integrity is quite limited.

In conclusion, the solution is a bit better than normal encryption, but not much.

iv. Complexity

The complexity of this solution is quite high in terms of client deploys. Since only a very limited set of operations can be performed on the data after encryption, a clear decision must be made as to which data can be protected with this method, i.e. which data will only need that set of operations and nothing more.

This solution also involves setting up a separate server that works as an external processor, which might not be attractive to a client.

The external processor protects the data, but that same data is only processed on the Splunk side, meaning that new Splunk commands need to be implemented to concatenate strings and perform mathematical operations at search time.

Finally, if needed, a command will also be used to decrypt the data for users with higher privileges, in which case public keys will need to be shared between all the involved servers. Maintaining public and private keys on clustered environments exponentiates the complexity of the solution even more.

f. GDPR analysis

Regarding GDPR, this prototype complies with everything that is stated in the regulation. All personal data is protected with the use of encryption - like said on GDPR's article 25 (General Data Protection Regulation - Final text neatly arranged, n.d.), "Data protection by default and by design", of section 2.1.4 - before it is stored on the indexer machine.

Also, the integrity of the data is maintained - like said on GDPR's article 5 (General Data Protection Regulation - Final text neatly arranged, n.d.), "Processing of personal data", shown on section 2.1.2 - since a user with enough privileges can still use a decryption command to see data in its original form. The only kind of data processing that can be done is to concatenate strings at search time and mathematic operations, which will still not affect the actual data that is stored on the indexer anyway.

g. Prototype conclusion

While being fully compliant with GDPR, this prototype offers the possibility to still perform some operations on the personal data, which is essential for the solution at hand.

Nevertheless, the really poor performance of this prototype completely throws it down the drain due to being impractical in an operation intelligence environment where scalability of data is to be expected. Having this in mind, the 6,6 hours of personal data protection by day can quickly grow to half a day or, in more extreme situations, to even more than a day, creating a bottleneck to all processes that surround this single operation.

Section 4.2.4 introduces SEAL which, similarly to Paillier, is a homomorphic encryption mechanism. Being a solution that was implemented by a team of developers from Microsoft, SEAL is expected to perform much better than Paillier both in performance and in the number of operations upon encrypted data that it provides.

4.2.4 Prototype 4 – SEAL library

The Simple Encrypted Arithmetic Library (SEAL) is a C++ homomorphic encryption library that was developed at Microsoft Research by a group of researchers in the Cryptography Research Group (Laine, 2017).

a. SEAL introduction

The SEAL's purpose is to be easy to use by both experts and non-experts of the homomorphic encryption field, by being as well-engineered and documented as possible.

The two basic homomorphic operations are additions and multiplications, of which additions can generally be thought of as being nearly free in terms of noise budget consumption compared to multiplications.

The SEAL algorithm requires 4 encryption parameters in order to be properly set:

- poly_modulus – a polynomial $x^n + 1$, where n is a power of 2;
- coeff_modulus - an integer modulus q which is constructed as a product of multiple distinct primes;
- plain_modulus – an integer modulus t;
- noise_standard_deviation – a standard deviation σ, which is assigned a standard value of 3,19;

SEAL cannot perform arbitrary computations on encrypted data because data becomes too corrupted to be decrypted after a certain number of operations is performed. This is determined by a specific quantity named "invariant noise budget" measured in bits. This noise budget and the rate at which it is consumed is determined by the above encryption parameters. Decryption becomes impossible when the noise budget reaches zero.

A simple life cycle of operations on the data is as follows:

1. Define the encryption parameters.
2. Construct a SEAL context object, which checks the validity and properties of the parameters that were just set.

3. Define an encoding scheme to represent the values to encrypt as polynomials. The values can be encoded with an IntegerEncoder, FractionalEncoder or PolyCRTBuilder. For the purposes of this thesis, only the first two are considered.
4. Generate the public and secret keys with a built-in key generator.
5. Define an encryptor object with the context and the public key.
6. Define a decryptor object with the context and secret key.
7. Encode the involved values with the defined built-in encoding scheme.
8. Encrypt the encoded values.
9. Perform the needed operations with built-in methods.
10. Calculate the noise budget of the encrypted data.
11. Decrypt the resulting data if the noise budget is still bigger than zero bits.

IntegerEncoder

A base-b expansion of the integer is computed, which uses a balanced set of representatives of integers modulo b as the coefficients.When b is odd the coefficients are integers between *–(b-1)/2* and *(b-1)/2*. When b is even, the integers are between *-b/2* and *(b-1)/2*, except when b is two and the usual binary expansion is used (coefficients 0 and 1).

Example with b=2:

26 = $2^4 + 2^3 + 2^1$ is encoded as the polynomial $1x^4 + 1x^3 + 1x^1$.

Example with b=3:

26 = $3^3 - 3^0$ is encoded as the polynomial $1x^3 - 1$

FractionalEncoder

Encodes fixed-precision rational numbers.

Expands the number in a given base b, possibly truncating an infinite fractional part to finite precision.

Example with b=2:

26.75 = $2^4 + 2^3 + 2^1 + 2^{-1} + 2^{-2}$ is encoded as $-1x^{1023} - 1x^{1022} + x^4 + 1x^3 + 1x^1$.

b. Prototype architecture

Two architectures were considered for this prototype.

First, an architecture with an external processor located between 2 Splunk forwarders, like shown on Figure 4.16. This external processor is where the SEAL algorithm is executed to protect the sensitive personal data before it reaches the Splunk indexer. The first forwarder sends the raw data to the external processor. The external processor then protects the data and stores it locally on a folder that is being monitored by another forwarder located on the same machine.

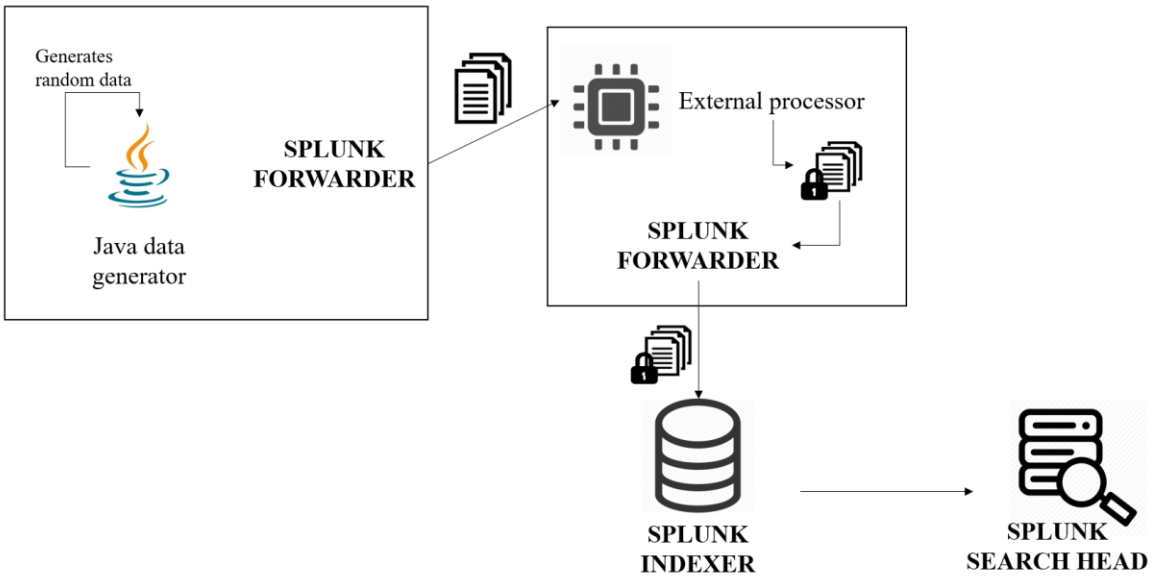

Figure 4.16 – SEAL's prototype architecture 1

Second, an architecture with an external processor located between the Splunk forwarder and the Splunk Indexer, like shown on Figure 4.17. This is better both performance and structure wise, since only 1 forwarder machine is needed, thus reducing the overhead and the installation requirements. On the other hand, it is much more difficult to implement because Splunk doesn't provide a communication API for C++.

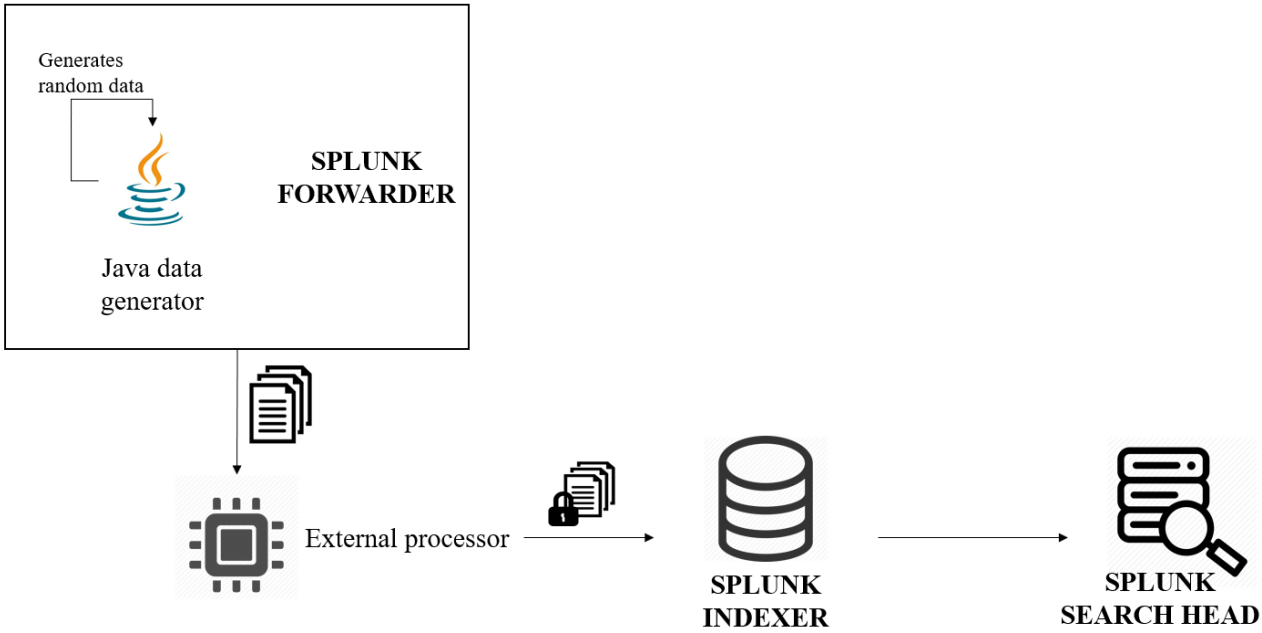

Figure 4.17 - SEAL's prototype architecture 2

c. Data protection

The data is protected on an external processor, before reaching Splunk territory. For the purposes of this prototype, all personal data was protected with the SEAL algorithm, in order to get the full impact of this solution.

The following 2 types of protection were tested with SEAL:

1. **Encryption with integer encoding**, along with appropriate mathematic operations. This type of encryption is used only on numeric fields, like “IdentificationNumber” for example;
2. **Encryption of strings**, by converting the string to Unicode, similarly to what was done on the concatenation adaption for the Paillier algorithm (see section 4.2.3).

Figure 4.18 shows an event with data that was already protected with SEAL. Due to the high complexity of the decrypted values that result from the SEAL algorithm, only the Address field was encrypted, in order to make the figure more visible.

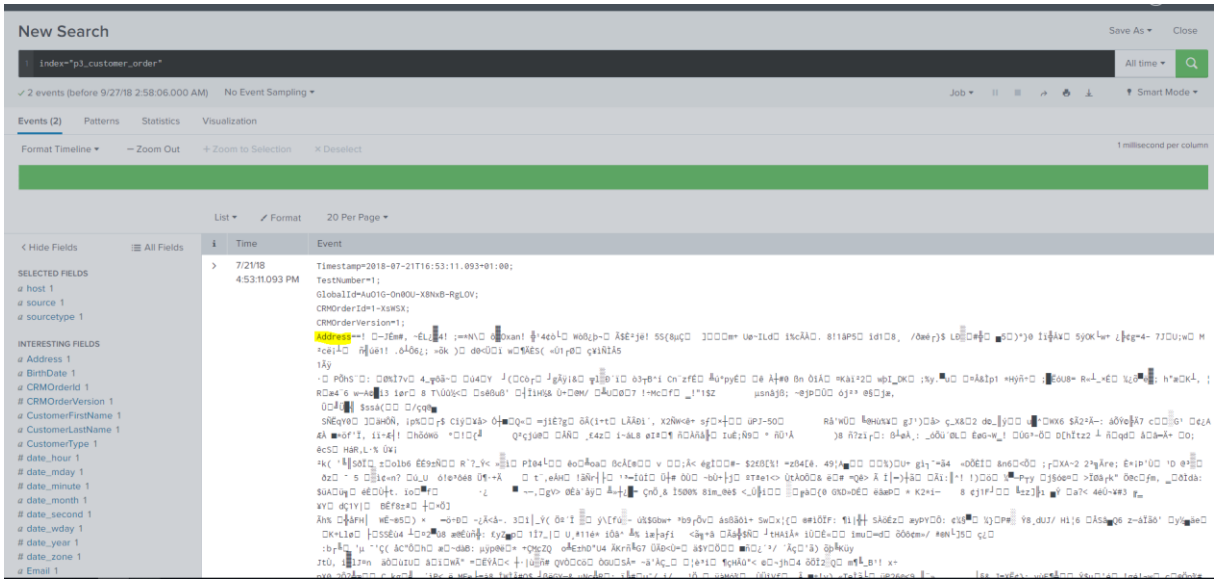

Figure 4.18 - SEAL's data protection

All fields can later on be decrypted by users with enough permissions with python commands on the Splunk side that invoke the C++ executables via command line/terminal.

For this purpose, a command called “decryptseal” was created. Similarly to the “decryptpaillier” command that was created for the Paillier prototype, this command receives field names as arguments and decrypts them using the public and private keys from the SEAL algorithm that are shared between the indexer and the external processor.

Figure 4.18 already shows what it looks like to encrypt the Address field with SEAL. Figure 4.19 shows the application of the “decryptseal” command into the encrypted Address field using the Administrator account.

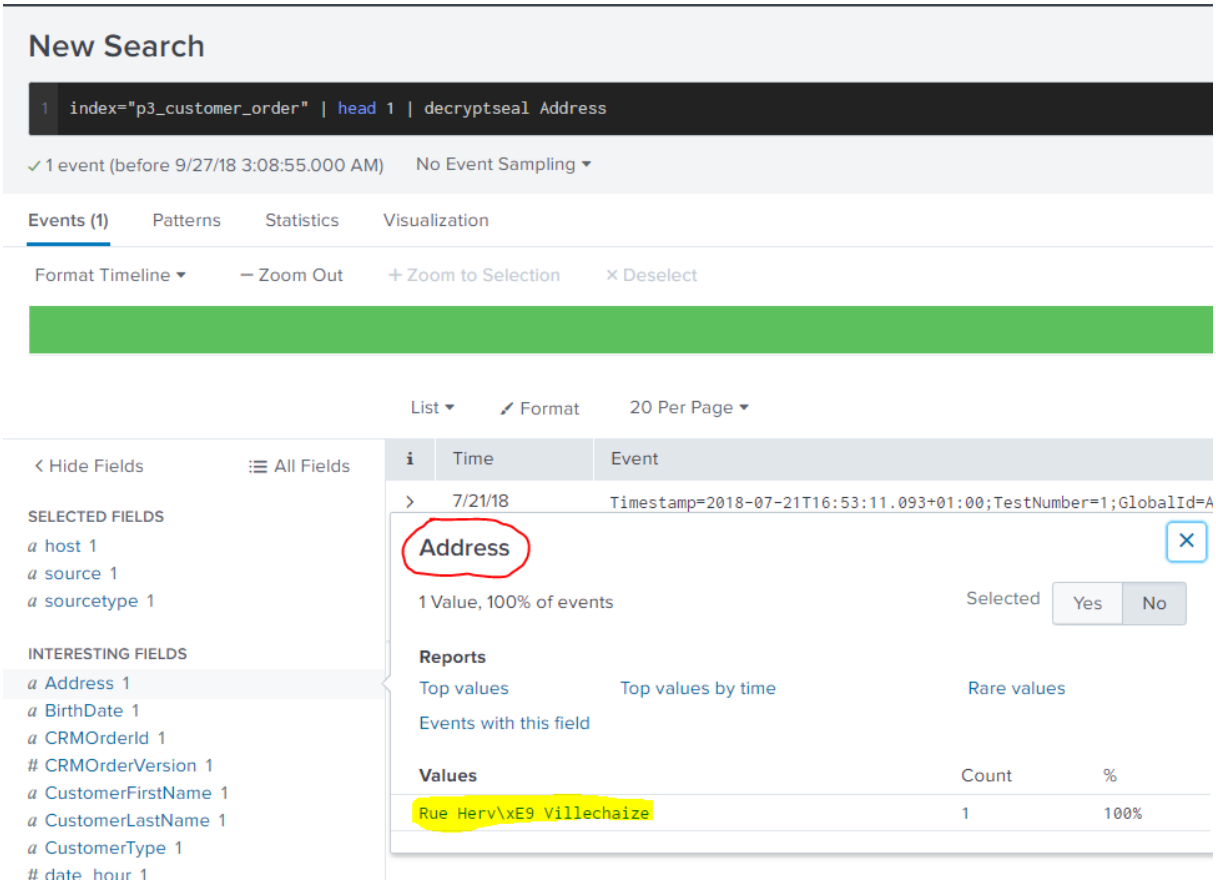

Figure 4.19 – SEAL field decryption

i. Integer encoding

The following course of action was taken for this scenario:

1. Two numbers, 5 and -7, were chosen and encoded using an Integer encoder;
2. The encoded values were encrypted;
3. Encrypted 5 was negated using the "negate" built in method;
4. The sum of the encrypted -5 with the encrypted -7 was calculated using the "add" built in method;
5. The result of the sum was multiplied by the encrypted -7 using the "multiply" built in method;
6. The final result was decrypted and then decoded with the Integer encoder;.

It's also worth noting that the invariant noise budget was calculated after each encryption step (encrypt, negate, add and multiply).

The following parameters were used:

Parameter	Value
poly_modulus	1x^2048 + 1
coeff_modulus	An 128-bit integer modulus with a degree of 2048.
plain_modulus	1 << 8
noise_standard_deviation	Standard value (3,19).

Table 4.8 - SEAL integer encoding's parameters

Three attempts were performed to reach the following performance data:

Step	1st attempt	2nd attempt	3rd attempt	Average
Key generation	6.878 milliseconds	7,395 milliseconds	7,536 milliseconds	7,269 milliseconds
Encoding of 5	0,077 milliseconds	0,009 milliseconds	0,029 milliseconds	0,038 milliseconds
Encoding of -7	0,470 milliseconds	0,678 milliseconds	0,489 milliseconds	0,546 milliseconds
Encryption of 5	10,883 milliseconds	10,785 milliseconds	14,696 milliseconds	12,121 milliseconds
Encryption of -7	21,130 milliseconds	21,504 milliseconds	32,153 milliseconds	24,929 milliseconds
Negate of 5	0,034 milliseconds	0,052 milliseconds	0,039 milliseconds	0,042 milliseconds
Sum of -5 and -7	0,061 milliseconds	0,057 milliseconds	0,081 milliseconds	0,066 milliseconds
Multiplication of -12 with -7	114,053 milliseconds	106,404 milliseconds	132,802 milliseconds	117,753 milliseconds
Decryption of the result	11,445 milliseconds	10,422 milliseconds	11,026 milliseconds	10,964 milliseconds

Table 4.9 - SEAL's data protection metrics

In all of the attempts, the noise started at 35 bits being only changed to 17 bits on the multiplication step.

ii. String

Similarly to what was done on Paillier, the following logic was implemented to simulate string concatenations, with a twist:

1. Both strings s1 and s2 are converted to Unicode;
2. Both unicodes U(s1) and U(s2) are encrypted using the Encryptor object from the SEAL library;
3. The encrypted unicodes E(U(s1)) and E(U(s2)) are concatenated with a secret split character between them.

For decryptions:

1. The encrypted data E(U(s1)) ⊕ E(U(s2)) is split by the secret split character that was previously defined, originating two different encrypted strings E(U(s1)) and E(U(s2)).
2. Both encrypted strings are decrypted using the Decryptor object from the SEAL library, thus originating the original Unicode values U(s1) and U(s2)
3. The Unicode values are concatenated back originating U(s1⊕ s2)
4. The Unicode value U(s1⊕ s2) is converted back to String originating s1⊕ s2.

The twist being that in C++ it's not possible to encode integers that are bigger than 2147483647, which makes it impossible to use UTF-8 Unicode for Strings. For example, a small string like "Andre" is 065110100114101 in Unicode, which is already a lot bigger than the limit. In order to go around this issue, every time a string surpasses the maximum value of integers the following logic is executed:

1. The string is split into 2 or more strings (depending on size)
2. The resulting strings are encrypted separately;
3. The encrypted string are concatenated with the same split character as in the actual encryption operations.

The same strings that were used before on Paillier were here tested as well, for a direct comparison.

String 1	String 2	Encryption time (string 1)	Encryption time (string 2)
“Andre “	“Lourenco”	22,098 ms	34,479 ms
“Cristiano “	“Ronaldo”	38,188 ms	35,497 ms
“AaBbCcDdEeFfGgHh”	“IiJjKkLlMmNnOoPp”	78,132 ms	74,151 ms
“qwertyQWERTYqwerty”	“QWERTYqwerty QWERTYqwerty”	74,407 ms	99,747 ms
“Hello “	“World”	21,785 ms	21,456 ms
“221B Baker Street“	“, London, England”	67,443 ms	65,743 ms
“abdefghijklmnopqrstuvwxyz ”	“ABCDEFGHIJKLMN OP QRSTUVWXYZ”	110,416 ms	120,451 ms

Table 4.10 - SEAL's string concatenation adaptation execution times

d. Access control

Just like on Paillier, a Role-Based Access Control policy was applied to the decryption command that was implemented on Splunk. The command was configured to be readable by the admin user role only. The admin role here is equivalent to the “Administrator” role that was introduced on the Internal users of section 3.1.1.

The decryption was already shown on the previous section. Nevertheless, if a user outside the admin role were to try to use this command, Splunk acts as if the command doesn’t exist at all, like shown on Figure 4.20.

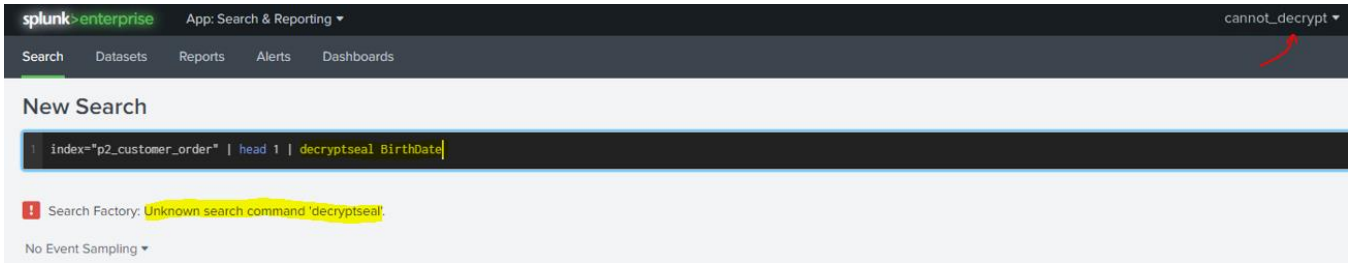

Figure 4.20 - SEAL's access control

e. Prototype analysis

This prototype is here analysed regarding performance, security, usability and complexity.

Just as on the Paillier prototype, the “External Processor approach was tested on this prototype, for the same reasons.

i. Performance

As was already shown on the Data Protection section, this prototype shows a massive improvement on the performance side when comparing with Paillier.

Table 4.11 proves the above statement with the following information:

- Both encryption start and end timestamps were sent to the Splunk indexer along with the already protected data.
- 3 tests were performed:
    - Test 1 – 1000 customer orders
    - Test 2 – 5000 customer orders
    - Test 3 – 10000 customer orders

Test #	Number of customer order (distinct global ids)	Number of events	Bytes	Total encryption time (secs)	Avg. time per event (secs)	Avg. time per MB (secs)
1	1000	10024	5 648 997	3 615,37	0,361	0,64
2	5000	50269	28 369 243	18 156,38	0,361	0,64
3	10000	100959	56 991 136	37 574,12	0.372	0.66

Table 4.11 - SEAL's performance metrics

As is shown on Table 4.11, this prototype offers a substantial improvement on the Performance metrics that were calculated for Paillier, while losing approximately one decimal value in comparison with the AES prototype. All things considered, SEAL encrypts all personal data of a single event in approximately 0,365 seconds and encrypts 1 MB in approximately 0,64 secs.

Taking into account the personal data estimate of 1% that was established on section 3.1.4 and considering that approximately 24GBs of data (~50 million events) are received every day, the OI application is expected to receive 240MBs of personal data per day.

In conclusion, 153,6 seconds or 2,56 minutes would be needed to encrypt all personal data of a single day, which is still very acceptable.

ii. Security

This solution can be applied on scenario 2 with combined encryption and anonymization, scenario 3 with pure encryption for specific data and scenario 5 with pure encryption for all data.

This solution is pretty good security wise, in the sense that the protected data (or portion of data) cannot be interpreted by the human eye without being decrypted, just like any normal encryption scenario.

iii. Usability

Even though the data is pseudonymized, it can still be processed for certain operations. Similary to Paillier, the amount of work that can be done on the data without jeopardizing its integrity is quite limited. Even though the deck of numeric operations is much more vast here (aside from add and multiply there is also negate, square, exponentiate, apply galois and many others), string manipulation is still limited to concatenations.

Table 4.12 illustrates the aforementioned facts.

Type of data	SEAL usage
Name	Can only be concatenated with other strings using the concatenation adaptation algorithm.
Age	All mathematic operations that are available on SEAL (egs.: Add, Subtract, Multiply, Divide, Negate, Power…) can be performed on this field.
Address	This field has both numeric values and strings.

	All the strings like the name of the street or the locality can only be concatenated with other strings using the concatenation adapatation algorithm. The numeric values like the street number can be processed with all the mathematic operations that are available on SEAL (egs.: Add, Subtract, Multiply, Divide, Negate, Power…).
Gender	Can only be concatenated with other strings using the concatenation adaptation algorithm.
Identification Number	All mathematic operations that are available on SEAL (egs.: Add, Subtract, Multiply, Divide, Negate, Power…) can be performed on this field.
Email	Can only be concatenated with other strings using the concatenation adaptation algorithm.
Phone Number	Even though this field is mostly numeric, it may have a prefix which is a string (eg.: "+351"). In case it hasn't, the prefix can be added with the concatenation adaptation algorithm. The remaining numeric value can be processed with all the mathematic operations that are available on SEAL (egs.: Add, Subtract, Multiply, Divide, Negate, Power…).
Birthdate	In case the birthdate is incomplete (eg.: doesn't have the year), it can be fixed with the concatenation adaptation algorithm. All the numeric values (day, month and year) can be processed with all the mathematic operations that are available on SEAL (egs.: Add, Subtract, Multiply, Divide, Negate, Power…).

Table 4.12 - SEAL usage by data type

In conclusion, the solution is a bit better than paillier and normal encryption in its usability, but not much.

iv. Complexity

The complexity of this solution is quite high in terms of client deploys. Since only a very limited set of operations can be performed on the data after encryption, a clear decision must be made as to which data can be protected with this method, i.e. which data will only need that set of operations and nothing more. Table 4.7 from the Usability analysis chapter already shows what kind of work can be performed for each type of data.

This solution also involves setting up a separate server that works as an external processor, which might not be attractive to a client.

The external processor protects the data, but that same data is only processed on the Splunk side, meaning that new Splunk commands need to be implemented to concatenate strings and perform mathematical operations at search time.

Finally, if needed, a command will also be needed to decrypt the data for users with higher privileges, in which case public keys will need to be shared between all the involved servers. Maintaining public and private keys on clustered environments exponentiates the complexity of the solution even more.

f. GDPR analysis

Just like Paillier, this prototype complies with everything that is stated in GDPR. All personal data is protected with the use of encryption - like said on GDPR's article 25 (General Data Protection

Regulation - Final text neatly arranged, n.d.), “Data protection by default and by design”, of section 2.1.4 - before it is stored on the indexer machine.

Also, the integrity of the data is maintained - like said on GDPR’s article 5 (General Data Protection Regulation - Final text neatly arranged, n.d.), “Processing of personal data”, shown on section 2.1.2 - since a user with enough privileges can still use a decryption command to see data in its original form. The only kind of data processing that can be done is to concatenate strings at search time and mathematic operations, which still doesn’t affect the actual data that is stored on the indexer anyway.

g. Prototype conclusion

Being a homomorphic encryption based prototype, SEAL is fully complaint with GDPR and at the same time allows some computation to be performed on the data without jeopardizing its security and integrity.

As predicted on the prototype conclusion of Paillier (see section 4.2.3) SEAL offers a huge improvement in terms of performance and the amount of computation/data processing it allows to be done is a lot bigger for numeric data, where it shines above all encryption schemes by offering a wide set of operations that can be performed without jeopardizing the security of the data.

Nevertheless, SEAL has exactly the same capabilities as Paillier for alpha-numeric data processing. Concatenation of strings is also possible with AES and with better performance, like shown on the performance analysis of section 4.2.2, thus making the use of homomorphic encryption on strings unjustifiable until a better solution comes up.

Section 4.3 proposes 2 different solutions that use both the SEAL and AES prototypes in order to use the best out of the two worlds while using different architectures to satisfy different client scenarios.

4.3 Proposed solutions

Two scenarios were here considered while proposing the solution:

1. Client has enough resources and installing an additional server is not a problem.
2. Client doesn’t have enough resources but is ok with having strict access control policies enforced onto all users of the platform.

It is also considered that it is not possible to have the best of both worlds, meaning that the client cannot ask for the best security with more open access control while having few resources, and vice-versa.

4.3.1 Solution 1 – External Processor based protection

This solution uses an External Processor to protect all personal data before it reaches the indexer and is to be used on the first scenario, where the client imposes no restrictions on the resources. This external processor will be executing one of two possible algorithms depending on the data format:

- If the data is in a numeric format, it is encrypted with the SEAL library. This library was already showed in use on section 4.2.4;
- If the data is in an alpha-numeric format, it is encrypted with the AES algorithm. This algorithm was already showed in use on section 4.2.2.

Once the data reaches the indexer machine, it is already protected and ready to be seen and processed by all users on the search head. For that effect, the following commands will need to be implemented on the Search Head:

- "decryptseal" – This command was already shown in use on the "Data Protection" section of the SEAL prototype (see section 4.2.4). Only its numeric variant will be used. It should be used only by administrators and should generate a log warning saying that the user "X" decrypted personal information of the event "Y". For this matter, the command should receive an additional parameter named "justification" where the user can put the reason of the decryption.
- "sealsum", "sealmultiply", "sealnegate", … - These commands were already showed in use on the "Data Protection" section of the SEAL prototype (see section 4.2.4). One command might be implemented per each operation that is provided by the SEAL library. They can be used by any user of the application.
- "decryptaes" – This command was already showed in use on the "Data Protection" section of the AES prototype (see section 4.2.2). It should be used only by administrators and should generate a log warning saying that the user "X" decrypted personal information of the event "Y". For this matter, the command should receive an additional parameter named "justification" where the user can put the reason of the decryption.

Once these commands are implemented, all users can perform the normal operations of an operational intelligence solution on the data.

Only the administrators will be able to decrypt the data. Even though it is not part of the scope of this project, the data decryption action should be stored on a local log file, so that it can later on be consulted. This is required by GDPR on article 30 "Records of processing activities" (General Data Protection Regulation - Final text neatly arranged, n.d.). As said on the mentioned article, the following information should be on the log file:

- Name and contact details of the controller;
- Purposes of the processing;
- Description of the categories of data subjects and their personal data;
- Categories of recipients to whom the personal data have been or will be disclosed, if applicable;
- Envisaged time limits for erasure, if possible and applicable;
- General description of the technical and organisational security measures.

It is strongly suggested that an email is also triggered with the justification for project responsible personnel.

4.3.2 Solution 2 – Search time protection

This solution encrypts the data on the search head server and is to be used on the second scenario, where the client has restrictions on quantity of resources but is ok with a more strict access control. Just like on the "External processor based protection" solution, two algorithms can be executed based on the time format:

- If the data is in a numeric format, it is encrypted with the SEAL library. This library was already showed in use on section 4.2.4;
- If the data is in an alpha-numeric format, it is encrypted with the AES algorithm. This algorithm was already showed in use on section 4.2.2.

In this case, the data reaches the indexer still in an unprotected format, meaning that it is still not ready to be shown on the search head. Thus, it needs an additional command to encrypt the data. The commands are as follows:

- "encryptseal" – This command was already shown in use on the "Data Protection" section of the SEAL prototype. Only its numeric variant will be used. It can be used by any user of the application.
- "decryptseal" – This command was already shown in use on the "Data Protection" section of the SEAL prototype (see section 4.2.4). Only its numeric variant will be used. It should be used only by administrators and should generate a log warning saying that the user "X" decrypted personal information of the event "Y". For this matter, the command should receive an additional parameter named "justification" where the user can put the reason of the decryption.
- "sealsum", "sealmultiply", "sealnegate", … - These commands were already showed in use on the "Data Protection" section of the SEAL prototype (see section 4.2.4).. One command might be implemented per each operation that is provided by the SEAL library. They can be used by any user of the application.
- "encryptaes" - This command was already showed in use on the "Data Protection" section of the AES prototype (see section 4.2.2). It can be used by any user of the application.
- "decryptaes" – This command was already showed in use on the "Data Protection" section of the AES prototype (see section 4.2.2). It should be used only by administrators and should generate a log warning saying that the user "X" decrypted personal information of the event "Y". For this matter, the command should receive an additional parameter named "justification" where the user can put the reason of the decryption.

Once these commands are implemented, all users can perform the normal operations of an operational intelligence solution on the data, having in mind that the encryption command should always be present on the beginning of each search.

Just like on the first solution, all decryption commands have the access limited to administrator users only and the execution of these commands should originate an entry on a log file to register o decrypted which event and why.

Still, since the data is kept unprotected on the indexer, the access to that server should be limited as follows:

- Can access the server: System Administrators and Administrators;
- Cannot access the server: Developers.

In order to centralize the access control of users to servers and to applications, it is recommended that an LDAP server is used to ease up the process. If the client doesn't have an LDAP server set up, then normal operative system's access control can be used for the server and the Splunk's built-in access control can be used for the application.

4.4 Summary

Four prototypes were suggested for the protection of operational intelligence data.

The first prototype is based on anonymization of data. This is by far the easiest solution to implement and deploy and the best performance wise. It is on its usability and GDPR compliance that it lacks the most. Once the data is anonymized it can no longer be used and it cannot be reverted back to its original state, making it impossible to be interpreted by operational intelligence solutions. Also, anonymizing data is the same as destroying data which, according to GDPR, goes against integrity requirements unless the affected individual applies his right to be forgotten (General Data Protection Regulation - Final text neatly arranged, n.d.). This solution should then be only used on those situations.

The second prototype makes use of the AES encryption scheme while limiting the access of users to the operations that decrypt the data based on their role. This prototype is easier to implement and deploy as part of a product because it doesn't require an additional server to be installed just for the data protection. Also, it offers even better performance than SEAL, making it much more desirable for scenarios with a lot of alpha-numeric data being processed. Its biggest downfall is that it lacks usability in the sense that the only operation that can be done on the protected data is to concatenate it with other data.

Finally, the last two prototypes make use of homomorphic encryption algorithms and showcase their capabilities in processing data that is already encrypted. Paillier is the weaker algorithm, allowing only add and multiply operations on numeric data and concatenations on alpha-numeric data. It was also proven that its performance is unacceptable for big data standards.

SEAL on the other hand is much more robust in terms of numeric data processing offering a much wider set of numeric operations to be performed. It also shows a big improvement in performance where it really shines with an average of approximately 0,002 seconds of encryption time per event. Still, it only offers the concatenation operation for alpha-numeric operations, where it is not justified to use it instead of normal and more common encryption schemes like AES.

Based on all the above prototypes, two final solution proposals were made using both homomorphic and symmetric encryption. These two proposals satisfy different client scenarios by offering different architectures.

The first proposed solution is based on the external processor approach and is well suited for clients with a good amount of resources where installing a new server is not a problem. It offers better compliance with GDPR due to the fact that the data is already stored in an encrypted format

The second proposed solution is based on protecting personal data at search time and suits better companies that have fewer resources but are ok with applying more strict access control techniques. Protecting data at search time means that the data is stored in cleartext and is only protected as it is read, thus offering weaker GDPR compliance. This is where the strict access control techniques come into the picture by offering a way of limiting the access to the data in cleartext.

Section 5 builds a conclusion for this thesis' report.

5 Conclusion

The arrival of GDPR and the lack of operational intelligence solutions that are compliant with it is seen as an opportunity to develop something that shines in the market. Also, the increasing popularity of homomorphic encryption is a good indicator that this should be the way to go when it comes to protecting data that still needs to be processed afterwards.

This thesis contributes to a real operational intelligence solution scenario implemented with Splunk, a product that provides means of searching, monitoring and analysing machine big data but is yet to have an adequate solution to protect personal data that is yet to come. For this matter, a set of performance, security, usability and complexity requirements was created and 4 different prototypes that use both Splunk external and internal libraries were considered.

The main objective of the thesis was to come up with an efficient method of protecting data while maintaining it usable by operational intelligence standards. This was partially achieved with homomorphic encryption where, for numeric data, the performance of the SEAL library was considerable. Nevertheless, it was also proven that there is still no efficient method to apply homomorphic encryption onto alpha-numeric data, making it mandatory to use the more common encryption methods like symmetric encryption with AES.

Finally, two well composed solutions were suggested for the problem.

The first solution uses an external processor that protects personal data before it reaches the Splunk environment. It proved to be efficient in terms of security and fully compliant with GDPR, while it lacked in the usability and complexity requirements due to requiring an additional server that is not attractive to clients.

The second solution only protects personal data at search time, meaning that it is stored on the indexer in a cleartext format. This is not compliant with GDPR unless there is a strict access control in the server that stores the data, which is a lot harder to achieve.

While the perfect solution would be somewhere between the two solutions, where the data is protected with a mixture of homomorphic and symmetric encryption algorithms at search time, such an approach is not possible with current state of the art in Splunk, meaning that there’s always going to be a compromise either in security or in usability.

In conclusion, the appropriate solution will always depend on the scenario at hand. If the client has less resources, one should choose the search time solution and enforce strict security and access control policies. If the client has more resources, one should choose the external processor solution which is compliant with all security and GDPR requirements.

6 References

Symcox, J. (2017, September 20). Nine out of ten firms not ready for GDPR. Retrieved from http://www.businesscloud.co.uk/news/nine-out-of-ten-firms-not-ready-for-gdpr.

Stageberg, J. (2017, September 13). Your Worst GDPR Nightmare, Unstructured Data. Retrieved from https://pt.slideshare.net/Dataversity/your-worst-gdpr-nightmare-unstructured-data.

Rutter Networking Technologies (2016, December, 27). What is Operational Intelligence? 5 Things to know about Splunk. Retrieved from https://www.rutter-net.com/blog/what-is-operational-intelligence-5-things-to-know-about-splunk. Last accessed on 2018, July.

Splunk. Splexicon: the Splunk glossary. Retrieved from https://docs.splunk.com/Splexicon. Last accessed on 2017, December.

Splunk (2017, December). Manuals, Splunk® Enterprise. Retrieved from http://docs.splunk.com/Documentation/Splunk/7.0.1/. Last accessed on 2017, December.

Splunk, Angelo Brancato, Dirk Nitschke (2017, September, 28). .conf2017 Data Obfuscation and Field Protection in Splunk. Retrieved from https://conf.splunk.com/files/2017/slides/data-obfuscation-and-field-protection-in-splunk.pdf. Last accessed on 2018, July.

Intersoft Consulting. General Data Protection Regulation. Final text neatly arranged. Retrieved from https://gdpr-info.eu/. Last accessed on 2017, December.

Almehmadi, A. (2015). On the Potential of Intent-based Access Control (IBAC) in Preventing Insider Threats.

Stallings, W. & Brown, L. (2012). Computer Security: Principles and Practice, 2nd Edition. Pearson Education.

Splunk (2017). A Layman's guide on how to operate your SIEM under the GDPR.

Craig Gentry (September, 2009). "A fully homomorphic e encryption scheme".

Xun Yi, Russell Paulet, Elisa Bertino (2014). "Homomorphic Encryption and Applications", Springer.

Laine, Kim. Microsoft Research, WA, USA (2017). "Simple Encrypted Aithmetic Library 2.3.1". Retrieved from https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf. Last accessed on August, 2018.

Judith Hurwitz, Robin Bloor, Marcia Kaufman, Dr. Fern Halper (2009). "Service Oriented Architecture for Dummies", 2nd IBM Limited Edition.

David Chappell (June, 2009). Enterprise Service Bus, O'Reilly Media.

Annex A – Regular expressions

The following regular expressions were used throughout this project to identify personal data.

Phone number:

((\+351|00351) (91|92|93|96)\d{7})|((\+33|0033) (01|02|03|04|05)\d{8})|((\+44|0033) (((01|02|03|05|07|08|09)\d{9})|(011\d{8})|(0800\d{7})|((013873|015242|015394|015395|015396|016973|016974|016977|017683|017684|017687|019467)\d{5})))

Company name:

^[a-zA-Z\u00C0-\u017F´\-\(\)]+(([',. -][a-zA-Z\u00C0-\u017F´\-\(\)])?[a-zA-Z\u00C0-\u017F´\-\(\)]*)*$

First name

^[a-zA-Z\u00C0-\u017F´]+(([',. -][a-zA-Z\u00C0-\u017F´])?[a-zA-Z\u00C0-\u017F´]*)*$

Last name

^[a-zA-Z\u00C0-\u017F´]+(([',. -][a-zA-Z\u00C0-\u017F´])?[a-zA-Z\u00C0-\u017F´]*)*$

Address

.+?(?=,), ([0-9]+), (-2|-1|RC|0|1|2|3|4|5|6|7|8|9|10|11|12)(Esq|Dir|Esquerdo|Direito|L|R|Left|Right|A|B|C|D|E|F), (\d\d\d\d\d|\d\d\d\d-\d\d\d|[A-Z][A-Z]\d\d \d[A-Z][A-Z]) - .+?(?=,), .+?(?=,), (PT|UK|FR)

Identification number

\d{9}

Email

.+?(?=@)@.+?(?=\.).(pt|com|iol\.pt)

Birthdate

([12]\d{3}-(0[1-9]|1[0-2])-(0[1-9]|[12]\d|3[01]))

Gender

MALE|FEMALE